SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Danijel Gotovac

Zagreb, 2019. godina.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Prof. dr. sc. Danijel Pavković, dipl. ing.

Student:

Danijel Gotovac

Zagreb, 2019. godina.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenom literaturom.

Zahvaljujem se svom mentoru, prof. dr. sc. Danijelu Pavkoviću na uloženom trudu, podršci, usmjeravanju u radu te ugodnoj suradnji pri izradi diplomskog rada. Također bi se zahvalio kolegama s fakulteta s kojima je uvijek bilo ugodno odrađivati fakultetske obaveze.

Posebno bih se htio zahvaliti svojoj supruzi Majdi, obitelji i rodbini na velikoj podršci tijekom studija.

Danijel Gotovac

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa:	
Ur. broj:	

DIPLOMSKI ZADATAK

Student: **DANIJEL GOTOVAC** Mat. br.: 0035170734

Naslov rada na hrvatskom jeziku: **Projektiranje i implementacija proporcionalno-integracijsko-derivacijskog regulatora za proces s aperiodskom dinamikom**

Naslov rada na engleskom jeziku: **Design and implementation of proportional-integral-derivative controller for a process with aperiodic dynamics**

Opis zadatka:

U radu pristupnik mora uraditi sljedeće:
1. Opisati sustave programibilnih logičkih kontrolera (PLC-a) Simatic S7-200.
2. Predložiti postupak sinteze proporcionalno-integracijsko-derivacijskog (PID) regulatora koji se temelji na vremenski-kontinuiranom modelu objekta upravljanja (procesa) dobivenog postupkom identifikacije dinamičkog modela procesa aperiodskog tipa.
3. Implementirati PID regulator i odabrani model objekta upravljanja u programskom okruženju Matlab/Simulink, izvesti izraze za parametre regulatora temeljene na približnom modelu procesa dobivenom postupkom identifikacije zasnovanom na skokovitoj pobudi na ulazu u proces.
4. Provesti simulacijsku analizu regulacijskog kruga s projektiranim regulatorom iz točke 3.
5. Implementirati PID regulator unutar CPU jedinice PLC-a koji će regulirati izlaznu veličinu procesa s aperiodskom dinamikom, a čije se ponašanje emulira u PLC-u primjenom odgovarajućih programskih rutina (simulatora procesa u realnom vremenu).
6. Snimiti odzive reference i regulirane veličine unutar PLC sustava u radu regulatora za zadani model procesa.
U radu je također potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
17. siječnja 2019.

Rok predaje rada:
21. ožujka 2019.

Predviđeni datum obrane:
27. ožujka 2019.
28. ožujka 2019.
29. ožujka 2019.

Zadatak zadao:

prof. dr. sc. Danijel Pavković

Predsjednica Povjerenstva:

prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
a_m	-	Parametri karakterističnog polinoma
D_i	-	Karakteristični odnosi
e	-	Stacionarna pogreška (PLC)
e_n	-	Stacionarna pogreška petlje u vremenu *n* (PLC)
e_{n-1}	-	Prethodna stacionarna pogreške u vremenu *n – 1* (PLC)
e_x	-	Stacionarna pogreška petlje u vremenu *x* (PLC)
K_C	-	Konstanta pojačanja propocionalnog člana (PLC)
K_D	-	Konstanta pojačanja derivacijskog člana (PLC)
K_I	-	Konstanta pojačanja integralnog člana (PLC)
K_p	-	Pojačanje PTn modela
K_R	-	Proporcionalno pojačanje regulatora
$M(t)$	-	Izlaz petlje u funkciji vremena (PLC)
$M_{initial}$	-	Inicijalna vrijednost izlaza petlje (PLC)
M_n	-	Izlaz petlje u vremenu *n* (PLC)
MD_n	-	Iznos derivacijskog člana u vremenu *n* (PLC)
MI_n	-	Iznos integracijskog člana u vremenu *n* (PLC)
MP_n	-	Iznos proporcionalnog člana u vremenu *n* (PLC)
MX	-	Integracijska suma (PLC)
n	-	Red sustava
PV_n	-	Iznos procesne veličine u vremenu uzorkovanja *n* (PLC)
PV_{n-1}	-	Iznos procesne veličine u vremenu uzorkovanja *n-1* (PLC)
SP_n	-	Iznos reference u vremenu uzorkovanja *n* (PLC)
t_r	s	Vrijeme prvog prolaska kroz stacionarno stanje
T_a	s	Vremenska konstanta prigušenja
T_e	s	Ekvivalentna vremenska konstanta
T_D	s	Derivacijska vremenska konstanta (PLC i opći slučaj)
T_d	s	Nadomjesno mrtvo vrijeme
T_I	s	Integracijska vremenska konstanta (PLC i opći slučaj)

T_p	s	Nadomjesna vremenska konstanta PTn modela
T_S	s	Vrijeme uzorkovanja (PLC)
T_t	s	Mrtvo vrijeme
w	-	Vanjski poremećaj
$y(s)$	-	Izlazna veličina petlje
$y_R(s)$	-	Referentna veličina na ulazu u petlju
σ_m	%	Koeficijent nadvišenja
SP_n	-	Iznos reference u vremenu uzorkovanja *n*

SAŽETAK

Regulacija temperature i protok fluida su često karakterizirani sporom aperiodskom dinamikom i mrtvim vremenom, i najčešće su modelirani uz pomoć sustava prvog reda s mrtvim vremenom (Engl. First-Order Process with Dead-Time, FOPDT). Da bismo dobili prikladniji modela procesa bez mrtvog vremena, FOPDT modela procesa se može aproksimirati vremenski-kontinuiranim PTn modelom procesa, karakteriziran istim nagibom (derivacija po vremenu) u točki infleksije aperiodskog odziva procesa. U ovom radu je predstavljena jedna od analitičkih metoda sinteze PID regulatora temeljena na vremenski-kontinuiranom modelu objekta upravljanja dobivenog postupkom identifikacije dinamičkog modela procesa aperiodskog tipa. Korištenjem PTn modela procesa kao osnove za sintezu sustava upravljanja, izvedeni su izravni analitički izrazi za parametre PID regulatora primjenom kriterija optimuma dvostrukog odnosa. Dobiveni parametri PID regulatora su uneseni u regulator implementiran u programibilnom logičkom kontroleru (PLC-u) te je provedena eksperimentalna verifikacija projektiranog regulatora.

Ključne riječi: PID regulator, PTn model procesa, optimum dvostrukog odnosa, identifikacija, PLC

SUMMARY

Heat and fluid flow processes are characterized by slow aperiodic dynamics and dead-time, which are frequently modeled by a first-order plus dead-time (FOPDT) process model. In order to obtain a more convenient dead-time free process model, the FOPDT process model can be aproximated with the more continuous-time PTn process model, characterized by the same response slope (time-derivative) at the flexion point. This master's thesis presents an analytical method of PID tuning based on time continuous process model identification comprising aperiodic dynamics. By utilizing the PTn process model as the basis for the control system design, straightforward analytical expressions have been derived for the PID controller parameters based on damping optimum criterion. Thus-obtained PID controller parameters were used within the programmable logic controller (PLC) based controller, which has been experimentally verified.

Key words: PID controller, PTn process model, damping optimum criterion, identification, PLC

1. Uvod

Od Prve industrijske revolucije kada su se koristile voda i para kao glavni izvori energije za mehanizaciju proizvodnje, kroz drugu revoluciju kada je počelo korištenje električne energije za stvaranje masovne proizvodnje, do treće industrijske revolucije kada su razvijene elektroničke komponente i informacijska tehnologija, pojavila se automatizacija proizvodnje. U početcima automatizacije koristila se relejna logika koja je uskoro zamijenjena znatno fleksibilnijim i pouzdanijim programibilnim logičkim kontrolerima (PLC). Kao i ostala računala, PLC je od svojih početaka do danas jako evoluirao i može izvršavati sve više funkcija potrebnih u proizvodnji. Između ostalog imaju mogućnost upravljanja i regulacije sustava pa tako nerijetko u sebi imaju integriran PID regulator.

Proporcionalno-integracijsko-derivacijski regulator (PID) je regulator sa tri člana koji ima dugu povijest u polju automatskog upravljanja. Zahvaljujući svojoj intuitivnosti i relativnoj jednostavnosti, uz zadovoljavajuće performanse koje može pružiti u širokom rasponu procesa, postao je standardni i najčešće korišteni regulator u industriji. Napretkom u tehnologiji evoluirao je i ovaj tip regulatora i sada se uglavnom implementira u digitalnoj formi. Moguće ga je pronaći u raznoj upravljačkoj opremi, bilo kao samostalni regulator ili kao dio programibilnog logičkog kontrolera (PLC-a).

Ovaj rad prikazuje jednu od metoda projektiranja i sinteze PID regulatora prikladnog za aperiodske procese višeg reda i usmjeren je postupku identifikacije zasnovanom na skokovitoj pobudi na ulazu u proces. Sinteza PID regulatora je bazirana na identifikaciji PTn modela procesa i aplikaciji optimuma dvostrukog odnosa, tako omogućavajući izravna algebarska pravila za podešavanje brzine odziva i prigušenja kod zatvorene petlje.

U poglavlju 2. kratko je opisana automatizacija i kad su se pojavili PLC-ovi. Također su navedene neke prednosti i nedostatci kod automatizacije, a u posljednjem potpoglavlju je opisana relejna logika. Poglavlje 3. daje uvod u PLC, opisuje njegovu ulogu i mjesto u proizvodnim procesima. Opisan je *hardware* i interna arhitektura sa svim bitnim dijelovima. Na kraju ovog poglavlja kratko su opisani *single box* izvedba i modularan tip PLC-a, te su nabrojane metode programiranja. U 4. poglavlju su opisane metode programiranja u ljestvičastom dijagramu i funkcionalna blok metoda, kao metode koje su najprikladnije za korisnike koji nemaju toliko iskustva s programskim jezicima. Na kraju ovog poglavlja su

kratko opisane i ostale metode programiranja PLC-a. Poglavlje 5. opisuje korišteni PLC tvrtke Siemens, model S7-200. Kratko je opisana njegova procesorska jedinica i ekspanzijski moduli te programski paket STEP7 – Micro/WIN. Opisan je njegov princip rada, način pristupanja podatcima, adresiranje lokalnih ulaza i izlaza, te elementi programa. Na kraju ovog poglavlja opisan je PID algoritam koji se koristi u S7-200 PLC-u. Poglavlje 6. opisuje projektiranje i sintezu PID regulatora dobivenog postupkom identifikacije dinamičkog modela procesa aperiodskog tipa, i korištenje optimuma dvostrukog odnosa za određivanje parametara regulatora. Navedeni parametri regulatora su implementirani u PID regulator na PLC-u i provedene su simulacije i usporedbe modela.

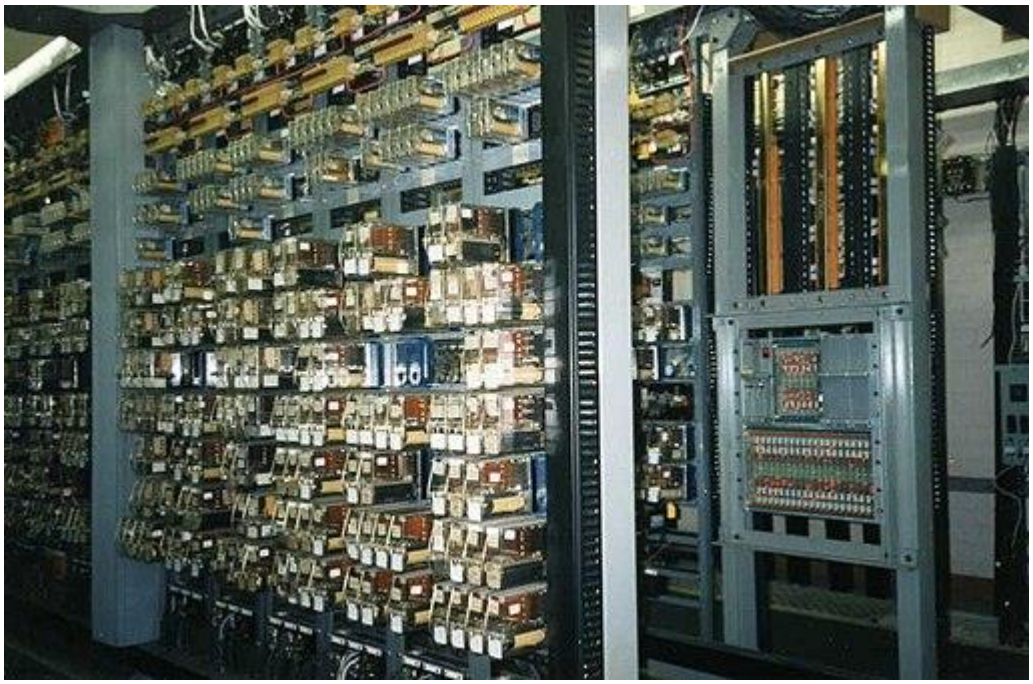
2. Automatizacija

Ljudska bića izrađuju stvari dugi niz godina. U osnovi većina proizvoda je rađena individualno po potrebi, ako je alat bio potreban, izrađen je rukom i iskorišten za izradu novih alata. Kako je vrijeme prolazilo, nove kompleksne tehnike su razvijene da bi pomogle ljudima u proizvodnim zadatcima. Metaloprerađivačka tehnologija, tkalački stanovi, mlinovi pogonjeni vodom, razvoj parnih generatora i motora s unutarnjim izgaradnjem su svi zajedno pripomogli većoj mogućnosti izrade raznih proizvoda, ali većina proizvoda je generalno izrađivana komadno od ljudi obučenih raznim tehnikama. Tek nakon industrijske revolucije i uporabom električne energije te mehanizama, masovna proizvodnja postaje uobičajena [1].

Automatizirani sustav je skupina uređaja koji rade zajedno da bi postigli zadatke ili proizveli proizvod ili skupinu proizvoda. Automobil je primjerice automatizirani sustav. Automobil ima računalo koje zaprima ulazne veličine sa raznih senzora i šalje izlazne veličine koji reguliraju rad motora i druge funkcije poput ABS-a, ESP-a i sl. [2]

Povijesno, mehanizacija je koristila ljudima za izvođenje fizičkih zahtjeva zadatka. Automatizacija, međutim, vodi mehanizaciju korak dalje, značajno reducirajući potrebu ljudskih senzornih i mentalnih zahtjeva dok simultano optimizira proizvodnju.

Vjeruje se da pojam automatizacija je prvi put koristio inženjer Ford Motor Company-a 40-ih godina prošlog stoljeća opisujući razne sisteme gdje su automatske akcije i kontrole zamijenile ljudski rad i inteligenciju. U to vrijeme kontrolni uređaj su bili elektromehanički po prirodi. Logika je izvođena s pomoću releja i timera isprepletnim ljudskim donošenjem odluka. Ožičenjem releja, timera, tastera, i mehaničkih senzora pozicija zajedno, jednostavne logičke sekvence su se mogle izvršavati uključivanjem i isključivanjem motora i aktuatora. [1]



Slika 1. Releji u automatiziranom sustavu [6]

Dolaskom računala i raznih elektroničkih komponenti, ovi kontrolni sustavi postaju manji, fleksiblniji, i jeftiniji za implementaciju i modificiranje. Na zahtjev General Motorsa, tvrtka Modicon je 70ih i 80ih godina prošlog stoljeća razvila programibilni logički kontroler kao zamjenu za dotrajalu relejnu logiku. Kako je tehnologija napredovala i više tvrtki koje se bave automatizacijom su došle na tržište, novi kontrolni proizvodi su razvijeni. Danas u industriji postoje mnogobrojne kontrolne naprave razvijene od stotina raznih proizvođača. [1]

Slika 2. PLC model Modicon 184 [6]

Na slici 2. je prikazan PLC model Modicon 184 za koji se govori da je zauvijek promijenio industriju [6].

2.1. Prednosti automatizacije

Neke od prednosti automatizacije:

- Ljudski operateri koji izvode teške fizičke i monotone poslove mogu biti zamijenjeni.
- Ljudi koji izvode zadatke u opasnim okolinama npr. temperaturni ekstremi, radioaktivne i toksične okoline, mogu biti zamijenjeni.
- Poslovi koji su iznad ljudskih mogućnosti su olakšani. Rukovanje s teškim ili velikim teretima, manipuliranje sitnim objektima, ili potreba za izradom proizvoda jako brzo ili jako sporo su primjer ovoga.
- Proizvodnja je često brža i troškovi radne snage su niži nego kod ekvivlantnih manualnih operacija.
- Kod automatiziranih sistema je moguće lagano ukomponirati provjere kvalitete i verifikacije kako bi se smanjio broj proizvoda koji ne zadovoljavaju ciljanu toleranciju.

- Automatizacija može poslužiti kao katalizator za unapređenje ekonomije poduzeća ili društva. Na primjer, bruto nacionalni dohodak i standard života u Njemačkoj i Japanu je dramatično povećan u 20-om stoljeću, velikim dijelom zbog uvođenja automatizacije u proizvodnje automobila, tekstila, oružja i drugih proizvoda za izvoz. [1]

2.2. Nedostatci automatizacije

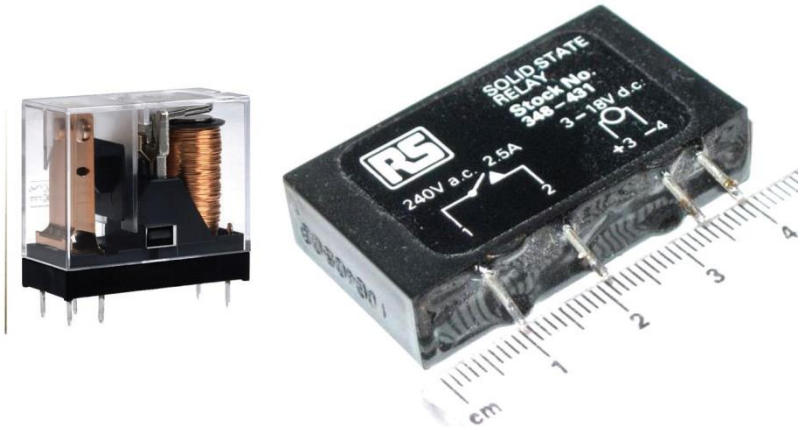
Neki od nedostataka automatizacije su:

- Trenutna tehnologija nije u mogućnosti automatizirati sve željene poslove. Neke poslove nije moguće lagano automatizirati, kao što je proizvodnja ili montaža proizvoda s nekonzistetnim veličinama komponenti ili kod poslova gdje je potrebna ručna spretnost. Neke stvari je najbolje ostaviti ljudskoj izradi i manipulaciji.
- Bilo bi skuplje automatizirati određene poslove nego ih izvoditi ručno. Automatizacija je najprikladnija za procese koji su ponovljivi, konzistentni, i velikog volumena.
- Trošak istraživanja i razvoja automatizacije za pojedini proces je teško precizno predvidjeti unaprijed. Obzirom na to da trošak može imati veliki utjecaj na profitabilnost, moguće je završiti automatiziranje procesa samo da bi otkrili kako ne postoji ekonomska prednost učinjenoga. Daljnjim razvojem različitih tipova proizvodnih linija moguće je preciznije estimirati benefit automatizacije.
- Inicijalni troškovi su relativno veliki. Automatizacija novog procesa zahtjeva goleme inicijalne investicije prema trošku jedne jedinice proizvoda.
- Odjel održavanja je potreban da bi servisirali i održavali automatizirani sustav da bi radio kako je predviđeno. Nedostatak pravilnog održavanja automatiziranog sustava će dovesti do gubitka proizvodnje i/ili pojave škarta.

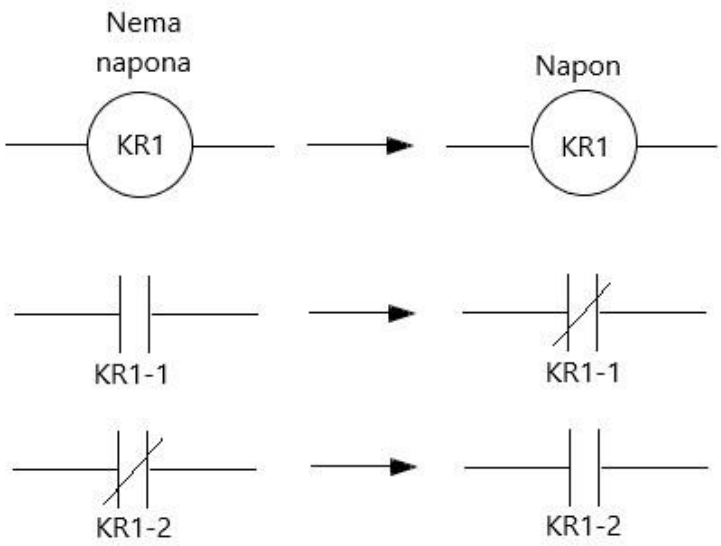
Globalno, prednosti imaju prevagu nad nedostatcima. Sa sigurnošću se može reći da zemlje koje su prihvatile automatizaciju uživaju veći standard života od onih koji nisu prihvatili u potpunosti. Istovremeno, postoji briga da automatizacijom mnogo ljudi gubi poslove koji su se do sad radili ručno. Bez obzira na socijalne implikacije, nema dvojbe da se produktivnost povećaje ispravnom aplikacijom tehnika automatizacije. [1]

2.3. Relejna logika

Relej je elektromehanički prekidač koji ima zavojnicu i set povezanih kontakata kao što je prikazano na slici 3. Kontakti mogu biti normalno otvoreni ili normalno zatvoreni. Kad se na zavojnicu spoji napon generira se elektromagnetno polje. Ovo elektromagnetno polje generira silu koja povlači kontakte releja, uzrokojući spajanje ili prekidanje spoja u eksternom strujnom krugu. Ove električki upravljane naprave se koriste u automobilskim i industrijskim aplikacijama da bi uključivale/isključivale uređaje velikih napona i struja. Iako postoji mogućnost upravljanja velikog industrijskog motora ili sustava paljenja direktno preko električnog kruga bez uporabe releja, taj izbor nije niti siguran niti praktičan. Primjerice, u tvornici, sklop za upravljanje motorom može biti smješten relativno daleko od visokonaponskog elektromotora i njegovog sustava napajanja iz sigurnosnih razloga. U tom slučaju, praktičnije je upravljanje uz pomoć releja nego direktna ožičenja sklopa za upravljanje s motorom i njegovim sustavom napajanja. [5]



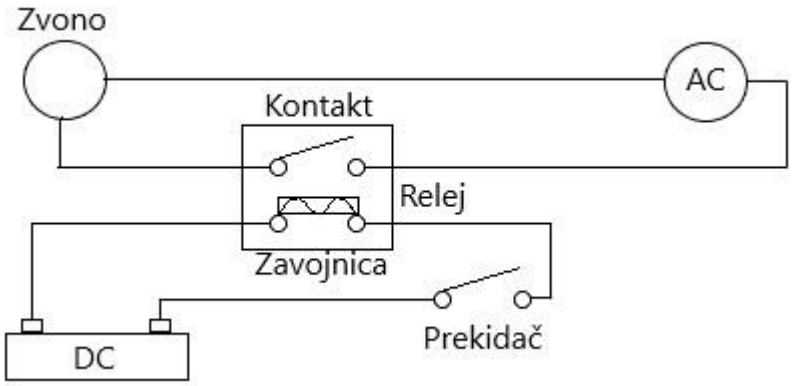
Slika 3. Tipični industrijski releji [5]



Slika 4. Relej s dva kontakta, normalno otvoren i normalno zatvoren [5]

Slika 4. Prikazuje kontrolni relej KR1 s dva kontakta: normalno otvoren (KR1-1) i normalno zatvoren (KR1-2). Na lijevoj strani slike, napon nije doveden do zavonice (KR1), i dva kontakta su u normalnom stanju. Na desnoj strani slike, napon je doveden do zavojnice i dva kontakta su u spoju.

Slika 5. Prikazuje jednostavni relejni krug za upravljanje zvonom koristeći jednopolni prekidač. Pritisak prekidača uzorkuje rad zvona. Relej se tipično korisiti za upravljanje uređajem za koji je potreban visoki napon ili velika struja. Relej omogućava uporavljanjem velike snage bez potrebe za mehaničkim prekidačem koji može podnositi veliku struju. Prekidač se obično koristi za upravljanje niskog napona ili struje, tj. upravljanje releja sa strane zavojnice. Na slici se mogu primjetiti dva odvojena kruga: donji krug koristi istosmjernu struju male snage, a gornji krug koristi izmjeničnu struju velike snage. Dva kruga su u spoju preko releja. Kao što je spomenuto, strana s istosmjernom strujom je spojena na zavojnicu, a strana s izmjeničnom strujom je odvojena od sklopa za upravljanje. Svaka strana je normalno upravljana sa svojim nezavisnim izvorom kod tipičnih industrijskih postrojenja. Naravno, u ovom primjeru nije isplativo zamijeniti relej s PLC-om, ali je u slučaju kad postoje stotine ili tisuće uređaja kojima treba upravljati. [5]



Slika 5. Jednostavni relejni krug [5]

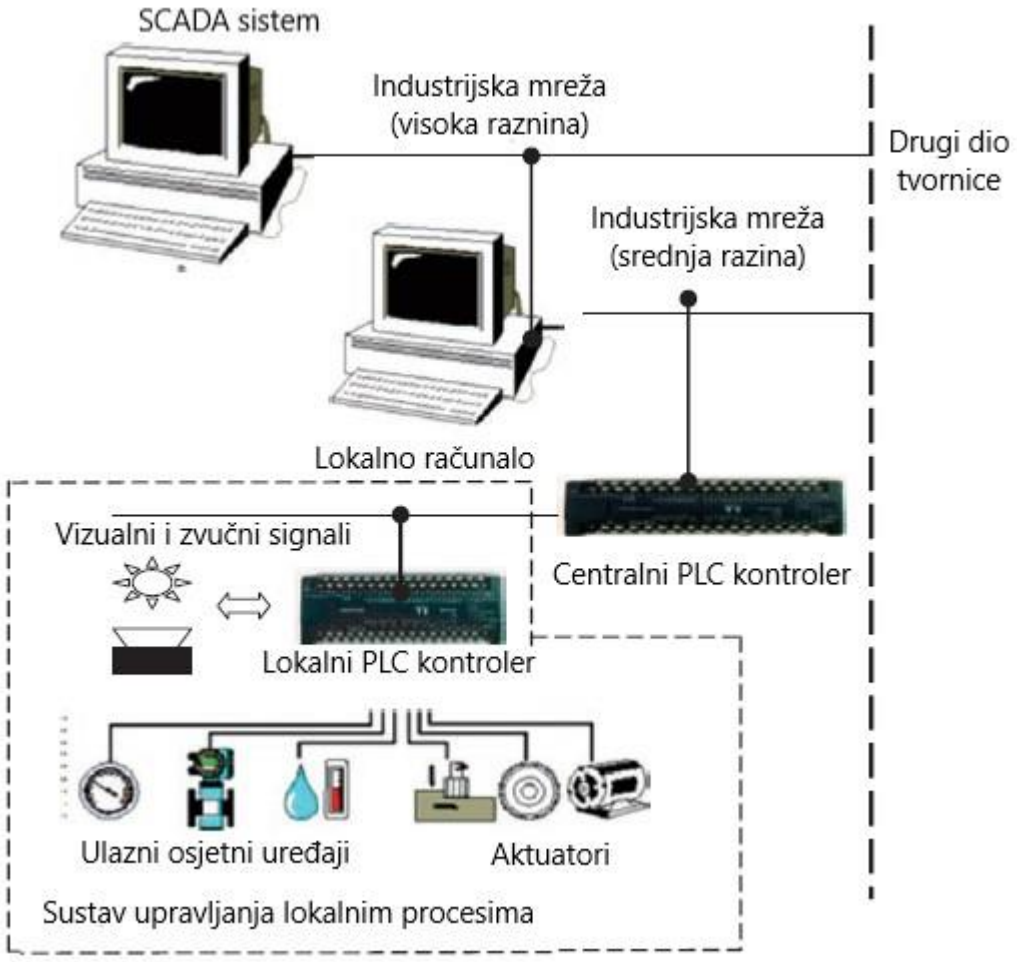
Relejni logički sustavi su kontrolne strukture prikladne za industrijsku i svakodnevnu uporabu. Operacije/procesi koji se upravljaju relejnim logičkim sustavima su fiksno ožičeni za razliku od programabilnih logičkih kontrolnih sistema. Ovi sistemi su ne-fleksibilni i postoje poteškoće u slučaju potrebe za preinakom nakon razvoja. Zbog toga što su funkcije relejnih logičkih kontrolera ugrađene u sam uređaj, lako je pronaći i otkloniti grešku ukoliko se problem ukaže. Takvi kotrolni sistemi su razvijeni s fiksnim karakteristikama za specifične aplikacije. Tipično, velike pumpe i motori će biti opremljeni s fiksno ožičenim relejima da bi bili zaštićeni od štete uzrokovane preotprećenjem i drugim neželjenim radnim uvjetima. PLC sustavi osiguravaju potrebnu fleksibilnost i omogućavaju buduća kontinuirana poboljšanja u procesu.

Releji se široko koriste u procesnom upravljanju i automatizaciji. PLC-ovi su jako prihvaćeni posljednjih 40 godina i zamijenili su većinu starih fiksno ožičenih relejnih kontrolnih sistema. Važno je razumijeti relejne kontrolne sustave da bi mogli cijeniti i prebaciti se na snažniju, lakšu za implementaciju, jeftiniju za održavanje, i pouzdaniju PLC kontrolu. [5]

3. Programibilni logički kontroleri

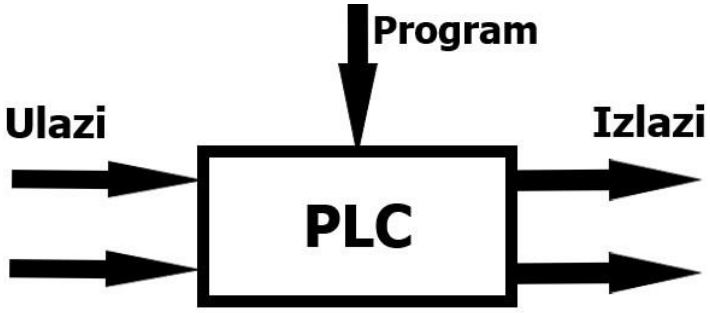
Razvoj programibilnih logičkih kontrolera (PLC) je primarno vođen zahtjevima proizvođača automobila koji su konstantno mijenjali kontrolne sisteme svoje proizvodne linije da bi se prilagodili novim modelima automobila. U prošlosti, za ovo je bilo potrebno reožičenje svih releja što je bio dugotrajan i skup postupak. Sedamdesetih godina prošlog stoljeća, pojavom poluvodičkih logičkih elemenata, nekoliko auto kompanija daje zadatak inženjerima da razviju način na koji bi bilo moguće zamijeniti kontrolnu logiku bez potrebe za potpunim reožičenjem sustava. Iz ove potrebe su evolvirali PLC-ovi. PLC-ovi su dizajnirani da bi bili razumljivi za korisnike kako bi električari mogli napraviti jednostavnu tranziciju od potpuno relejnog upravljanja do elektroničkih sistema. Oni daju korisnicima mogućnost prikazivanja i rješavanja problema *ladder* logike koja prikazuje logiku u stvarnom vremenu. Logiku je moguće programirati i testirati bez potrebe za spajanjem releja i žica.

PLC je računalo s jednom misijom. Obično nema monitor, tipkovnicu, i miš, jer je obično programiran da bi upravljao strojem ili sustavom koristeći jedan program. Operater na stroju ili u pogonu, rijetko, ako ikad, direktno radi na programu PLC-a. Kad je potrebno urediti ili napraviti PLC program, uglavnom se korisiti (ali ne uvijek) osobno računalo spojeno na njega. Podatcima s PLC-a se može pristupiti pomoću SCADA sistema i sučelja čovjek-stroj (*Human-Machine Interface – HMI*), da bi dobili grafičku reprezentaciju statusa pogona. Slika 6. je shematski prikaz kontrolne mreže PLC-a u industrijskim sustavima. [3]



Slika 6. Shematski prikaz kontrolne mreže PLC-a [3]

Programibilni logički kontroler je specijalan oblik mikroprocesorski baziranog kontrolera koji koristi programibilnu memoriju za spremanje instrukcija i za implementiranje funkcija kao što su logika, sekvenciranje, mjerenje vremena, brojanje, aritmetika, radi upravljanja strojevima i procesima. Dizajnirani su da bi njima upravljali inženjeri koji možda limitirano poznaju računalne sustave i računalne jezike. Nisu dizajnirani tako da samo računalni programeri mogu postaviti ili promijeniti programe. Prema tome, dizajneri PLC-a su ga preprogramirali da bi upravljački program mogli unijeti koristeći jednostavnu, intuitivnu formu jezika. Pojam logika se koristi jer se programiranje prije svega bavi s implementiranjem logike i komutacijskih operacija. Ulazne jedinice, pr. senzori i prekidači, i izlazne jedinice u sistemu koji se kontrolira, pr. motori, ventili, itd. su spojeni s PLC-om. Operater tad unese slijed uputa, pr. program, u memoriju PLC-a. Kontrolor tad „nadgleda“ ulaze i izlaze prema ovom programu i izvršava kontrolna pravila za koja je programiran.



Slika 7. Programibilni logički kontroler [4]

PLC-ovi imaju veliku prednost da isti osnovni kontroler može biti korišten u širokom rasponu kontrolnih sustava. Da bi modificirali kontrolni sistem i pravila po kojima će raditi, sve što je potrebno je da operater unese drukčiji set instrukcija. Nema potrebe za ponovnim ožičenjima. Rezultat je fleksibilni, isplativi, sistem koji može biti korišten s kontrolnim sistemima koji se razlikuju po svojoj prirodi i složenosti.

PLC-ovi su slični računalima ali dok su računala optimizirana za izračune i prikaz, PLC-ovi su optimizirani za upravljačke zadatke i industrijsku okolinu. Stoga, PLC-ovi su:

1. Robustni i dizajnirani su da bi podnosili vibracije, temperature, vlagu i buku.
2. Imaju sučelja za ulaze i izlaze unutar samog kontrolera.
3. Lako se programiraju i imaju lako razumljivi programski jezik s logičkim i komutacijskim operacijama.

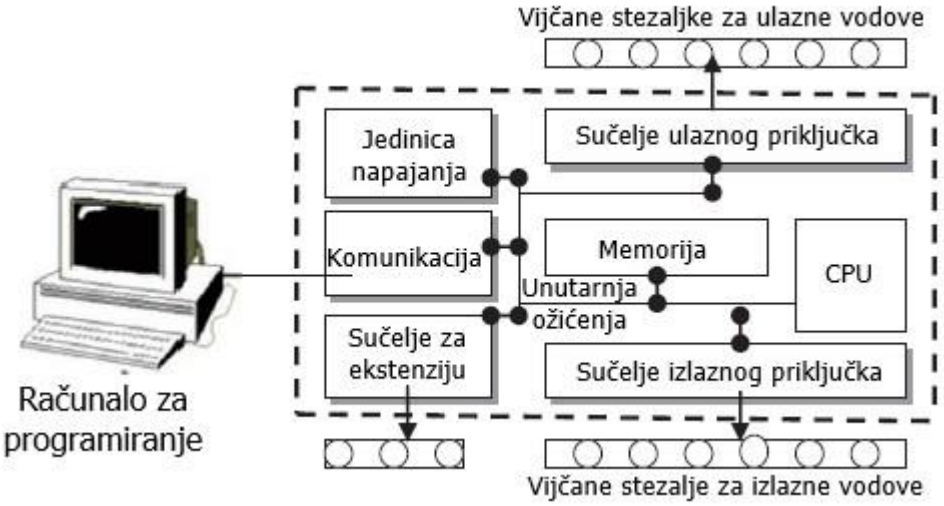
Prvi PLC je razvijen 1969. godine. Danas se naširoko koriste i imaju raspon od malih samostalnih jedinica za upotrebu s možda 20 digitalnih ulaza/izlaza do modularnih sistema koji se mogu koristiti za velike brojeve ulaza/izlaza, rade s digitalnim ulazima/izlazima, i izvršavaju PID regulaciju. [4]

Slika 8. Tipični industrijski PLC-ovi [5]

Od razvoja PLC-ova, stari i novi proizvođači su se natjecali da bi proizveli naprednije i jednostavnije za korištenje sisteme. Slika 8. Prikazuje primjer popularnih PLC-ova u industriji. Može se primjetiti raznovrsnost veličina i očito povezanih mogućnosti, a time i razlika u cijeni. Većina dobavljača omogućava integraciju drugih PLC-ova kao dio umreženog distributivnog sustava upravljanja. Moguće je implementirati ekstremno veliki sustav upravljanja na jedan PLC sustav s velikim brojem međusobno povezanih šasija i modula. [5]

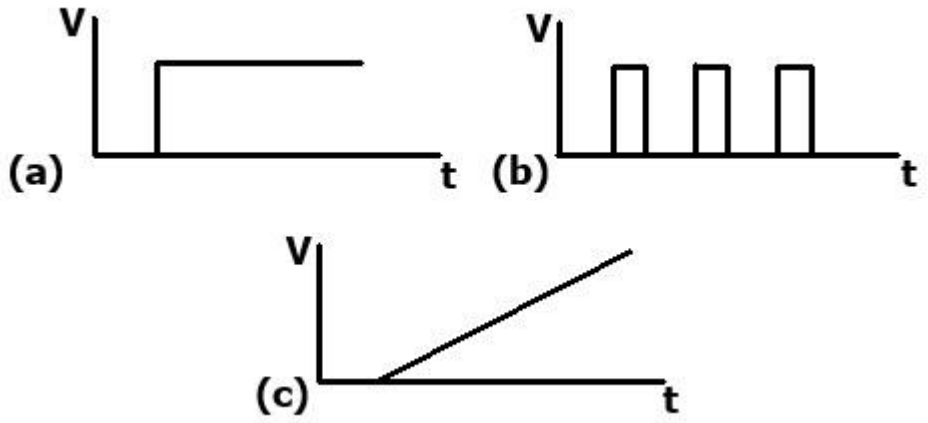
3.1. Hardware

Tipično PLC sistem ima osnovne funkcionalne komponente poput procesorske jedinice, memorije, jedinice napajanja, ulazno/izlaznog sučelja, komunikacijskog sučelja i uređaja za programiranje. Slika 9. pokazuje osnovni raspored.



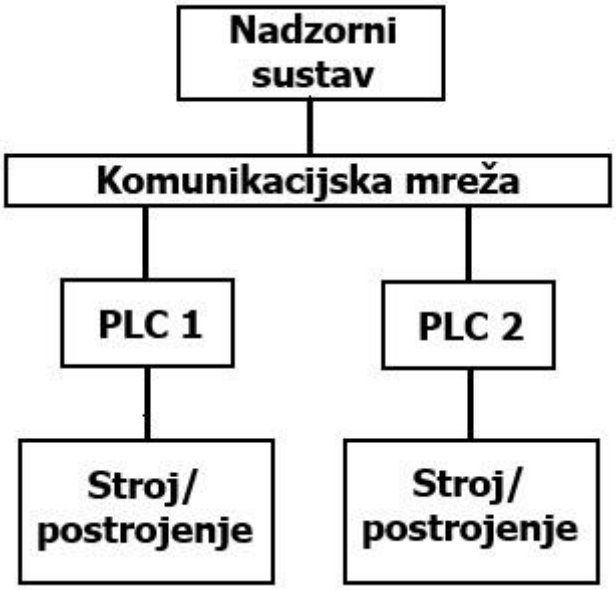
Slika 9. Osnovni elementi PLC-a [3]

1. Procesorska jedinica ili centralna procesorska jedinica (*CPU – Central processing unit*) je jedinica koja sadržava mikroprocesor i ona interpretira ulazne signale i sprovodi kontrolne akcije, prema programu spremljenom u njenoj memoriji, komunicirajući odluke kao akcijske signale prema izlazima.
2. Jedinica napajanja je potrebna za pretvorbu mrežne izmjenične struje u nisku istosmjernu struju (5 V) potrebnu za procesor i električne krugove u ulaznim/izlaznim modulima sučelja.
3. Računalo za programiranje se koristi za unos potrebnog programa u memoriju procesora. Program je razvijen u računalu i transferiran u memorijsku jedinicu PLC-a.
4. Memorijska jedinica je mjesto gdje je program spremljen i odakle će se koristiti za kontrolne akcije koje će provoditi mikroprocesor i tu su spremljeni podatci od ulaza za procesiranje i izlaza za izdavanje.
5. Ulazno/izlazne sekcije su mjesta gdje procesor prima informaciju od eksternih uređaja i komunicira informaciju prema eksternim uređajima. Pr. ulazi mogu biti od prekidača, temperaturnih senzora, senzora protoka i sl. Izlazi mogu biti razni ventili, motori i sl. Ulazni i izlazne naprave mogu biti klasificirane prema tipu signala poput: diskretni, digitalni i analogni. Kod diskretnih i digitalnih signala postoje dva stanja: uključen i isključen. Prema tome, prekidač je naprava koja daje diskretni signal, odnosno ima ili nema napona. Digitalne naprave su u naravi diskretne naprave koje daju sekvence on-off signala. Analogne naprave daju signale čija veličina je proporcionalna veličini promatrane varijable. Primjerice, temperaturni senzor može dati napon proporcionalan temperaturi.



Slika 10. Signali: (a) diskretni, (b) digitalni, (c) analogni [4]

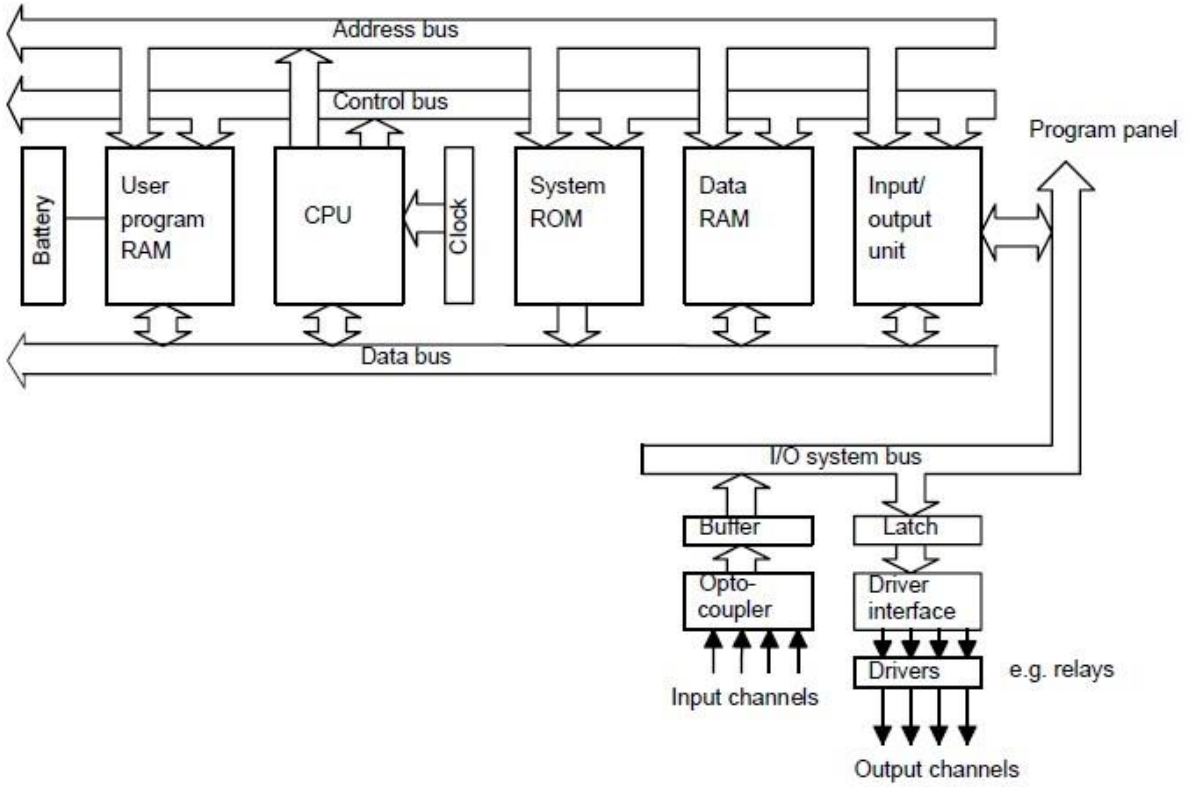
6. Komunikacijsko sučelje se koristi za primanje i slanje podataka na komunikacijsku mrežu od ili do drugih udaljenih PLC-ova. Bavi se akcijama poput verifikacije uređaja, akvizicije podataka, sinkronizacije između korisničkih aplikacija i upravljanja vezama.



Slika 11. Osnovni komunikacijski model [4]

3.2. Interna arhitektura

Slika 12. prikazuje osnovnu unutarnju arhitekturu PLC-a. Sastoji se od centralne procesorske jedinice (CPU) koja sadržava sistemski mikroprocesor, memoriju, ulazno/izlazne strujne krugove. CPU upravlja i obrađuje sve operacije unutar PLC-a. Sadrži generator radnog takta CPU jedinice tipično između 1 i 8 MHz. Ova frekvencija određuje radnu brzinu PLC-a i pruža *timing* i sinkronizaciju za sve elemente u sustavu. Informacija unutar PLC-a se prenosi pomoću digitalnih signala. Interni putovi kojima digitalni signali prolaze zovu se sabirnice. U fizičkom smislu, sabirnica je samo broj vodiča preko kojih električni signali prolaze. Mogu biti vodovi na tiskanim pločicama ili žice u *flat* kabelima. CPU koristi podatkovnu sabirnicu za slanje podataka između sastavnih elemenata, adresna sabirnica za slanje adresa lokacija za pristup pohranjenim podatcima i kontrolna sabirnica za signale vezane za upravljačke akcije. Sistemski sabirnica se korisiti za komunikaciju između ulazno/izlaznih portova i ulazno/izlazne jedinice.



Slika 12. Arhitektura PLC-a [4]

3.2.1. CPU – centralna procesorska jedinica

Interna struktura CPU-a ovisi o dotičnom mikroprocesoru. Generalno postoje:

1. Aritmetičko logička jedinica (*Arithmetic and logic unit – ALU*) koja je odgovorna za manipulaciju podataka i izvršavanje aritmetičkih operacija zbrajanja i oduzimanja te logičkih operacija I, ILI, NE i EX-ILI.
2. Memorija, tj. registri, koji se nalaze unutar mikroprocesora i koriste se za spremanje informacija vezanih za egzekuciju programa.
3. Kontrolna jedinica koja se koristi za kontrolu vremenskog izvođenja operacija.

3.2.2. Sabirnice

Sabirnice su putanje koje se koriste za komunikaciju unutar PLC-a. Informacija se prenosi u binarnom obliku, primerice kao grupa bitova gdje je bit binarna znamenka 1 ili 0, pr. on/off stanje. Termin *riječ (word)* se koristi za grupu bitova koji sadržavaju neku informaciju. Tako 8-bitna riječ može biti binarni broj 00100110. Svaki od bitova se komunicira simultano kroz svoju paralelnu žicu. Sistem ima četiri sabirnice*:*

1. Podatkovna sabirnica prenosi podatke unutar procesa koje daje CPU. Kad se za mikroprocesor kaže da je 8-bitni to znači da ima internu podatkovnu sabirnicu koji može raditi s 8-bitnim brojevima. U principu to znači da može izvoditi operacije između 8-bitnih brojeva i dati rezultate kao 8-bitne vrijednosti.
2. Adresna sabirnica se koristi za prijenos adresa memorijskih lokacija. Da bi svaka riječ mogla biti locirana u memoriji, svaka memorijska lokacija mora imati svoju jedinstvenu adresu. Lokacija svake riječi ima svoju adresu tako da podatci koji su spremljeni na pojedinoj lokaciji mogu biti pristupljeni od CPU-a ili za čitanje podataka s te lokacije, ili za snimanje novih podataka na toj lokaciji. Adresna sabirnica nosi informaciju kojoj adresi će se pristupiti. Ako adresna sabirnica sadržava 8 linija, broj 8-bitnih riječi, i stoga broj dinstinktnih adresa je $2^8 = 256$. Kod 16 adresnih linija, moguće je 65536 adresa.
3. Kontrolna sabirnica prenosi signale koje CPU koristi za upravljanje, pr. da bi informirao memorijske naprave bili primali podatke od ulaznih/izlaznih podataka i za prijenos vremenskih signala korištenih za akcije sinkroniziranja.
4. Sistemska sabirnica se koristi za komunikaciju između ulazno/izlaznih portova i ulaznih/izlaznih jedinica.

3.2.3. Memorija

Postoji par elemenata memorije u PLC sistemu:

1. Sistemska *read-only-memory (ROM)* na koju se permanentno sprema operacijski sustav i fiksni podatci korišteni od CPU-a.
2. *Random-access memory (RAM)* za korisnički program.
3. *Random-access memory (RAM)* za podatke. Ovdje se spremaju informacije o statusu ulaznih/izlaznih jedinica a vrijednosti *timer-a* i *counter-a* i drugih internih uređaja. Podatak RAM se nekad referira kao tablica podataka (*data table*) ili tablica registara (*register table*). Dio ove memorije, pr. blok adresa, će biti stavljen sa strane za ulazne i izlazne adrese i stanja tih ulaza i izlaza. Dio će biti stavljen sa strane za unaprijed postavljene podatke i dio za spremanje vrijednosti *counter-a, timer-a* i sl.
4. Postoji mogućnost dodatnog memorijskog modula, *EPROM – erasable and programmable read-only-memory* za ROM koju je moguće programirati i prema tome će program biti permanentan.

Programi i podatci u RAM-u se mogu mijenjati od korisnika. Svi PLC-ovi će imati određenu količinu RAM-a za spremanje programa koji su bili razvijeni od korisnika i za spremanje podataka programa. Ali, za sprečavanje gubitka programa dok je napajanje isključeno, koristi se baterija unutar samog PLC-a koja održava podatke RAM-a neko određeno vrijeme. Nakon što je program razvijen u RAM-u može ga se učitati u EPROM memorijski čip, često kao dodatni modul za PLC, i tako napraviti permanentnim. Dodatno postoji privremeni *buffer-i* za ulazne i izlazne kanale.

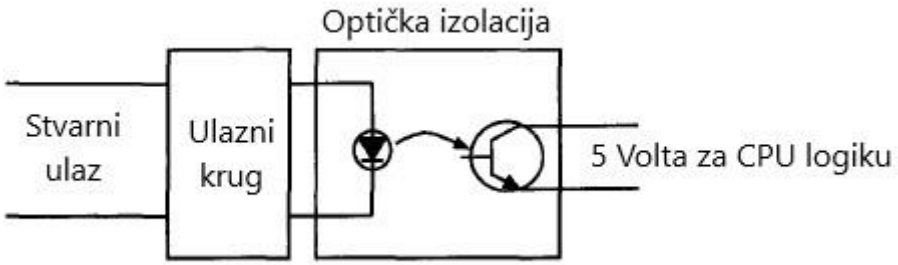
Kapacitet memorijske jedinice je određen brojem binarnih riječi koje može spremiti. Tako, ako je veličina memorije 256 riječi onda može spremiti 256 x 8 = 2048 bitova ako su korištene 8-bitne riječi, i 256 x 16 = 4096 bitova ako su korištene 16-bitne riječi. Veličine memorija su često specificirane brojem dostupnih mjesta za pohranu gdje bi primjerice 1K predstavljalo $2^{10} = 1024$. Proizvođači snabdijevaju memorijske čipove s mjestima za pohranu grupiranim u grupama od 1, 4 i 8 bitova. 4K x 1 memorija ima 4 x 1 x 1024 lokacije bitova. 4K x 8 memorija ima 4 x 8 x 1024 lokacije bitova. Termin bajt *(byte)* se koristi za riječ duljine 8 bitova. Tako 4K x 8 memorija ima 4096 bajtova. S 16-bitnim adresnim *bus-evima* možemo imati 2^{16} različitih adresa pa tako, s 8-bitnim riječima pohranjenim na svaku

adresu, možemo imati $2^{16} * 8$ mjesta za pohranu i tako koristiti memoriju veličine $2^{16} * \frac{8}{2^{10}} = 64K * 8$ što bi bilo oko četiri 16K x 8-bitnih memorijskih čipova.

3.2.4. Ulazno/izlazna jedinica

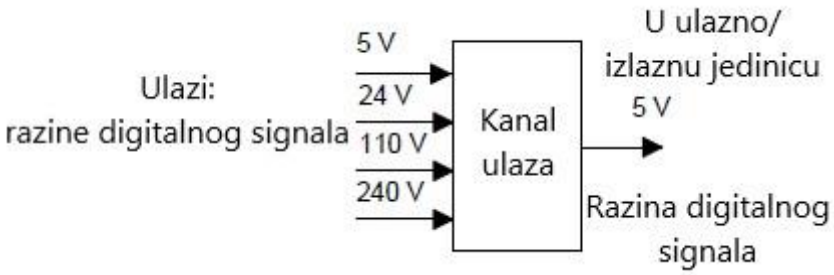
Ulazno/izlazna jedinica pruža sučelje između sistema i vanjskog svijeta, dozvoljavajući konekcije kroz ulazno/izlazne kanale do ulaznih naprava kao što su senzori i izlaznih naprava kao što su motori ili solenoidni ventili. Kroz ulazno/izlazne jedinice također unosimo programe preko programskog panela. Svaka ulazno/izlazna točka ima jedinstvenu adresu kojoj CPU može pristupiti.

Ulazno/izlazni kanali pružaju izolaciju i funkcionalnosti za prilagodbu signala tako da senzori i aktuatori mogu često biti spojeni na njih bez potrebe za dodatnim ožičenjima. Električna izolacija od vanjskog svijeta je često uz posredovanje optoizolatora (često se koristi pojam optokapler – *optocoupler*). Slika 13. prikazuje princip optoizolatora. Kad digitalni puls prođe kroz LED diodu, puls infracrvene radijacije se proizvede. Fototranzistor detektira ovaj puls i dovodi do rasta napona u tom krugu. Međuprostor između LED diode i fototranzistora daje električnu izolaciju ali digitalni puls iz jednoga kruga svejedno uzrokuje pojavu digitalnog pulsa u drugom krugu.



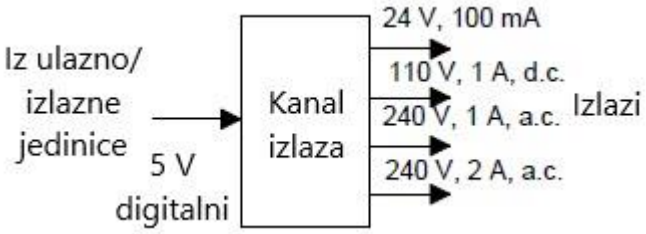
Slika 13. Optoizolator [2]

Digitalni signal koji je generalno kompatibilan s mikroprocesorom u PLC-u iznosi 5 volta. No ipak, kondicioniranje signala u ulaznom kanalu, s izolacijom, omogućuje širok raspon ulaznih signala koje može isporučiti. Kod većih PLC-ova raspon ulaza varira u naponu pa postoji 5 V, 24 V, 110 V, i 240 V digitalni/diskretni, tj. on/off signali. Manji PLC će vjerojatno imati samo jedan tip ulaza, pr. 24 V.



Slika 14. Razine ulaza [4]

Izlaz iz ulazno/izlazne jedinice će biti digitalni razine 5 V. Međutim, nakon kondicioniranja signala pomoću releja, tranzistora i trijaka, izlaz iz izlaznog kanala može biti 24 V, 100 mA sklopni signal; ili 1 A, 110 V istosmjerne struje; 1 A, 240 V izmjenične struje... Kod manjih PLC-ova svi izlazi mogu biti jednog tipa, pr. 240 V izmjenična struja, 1 A. Kod modularnog PLC-a moguć je odabir raspona izlaza ovisno o odabranim modulima koji će se koristiti.



Slika 15. Razine izlaza [4]

Izlazi su specificirani kao tipa releja, tipa tranzistora ili tipa trijaka:

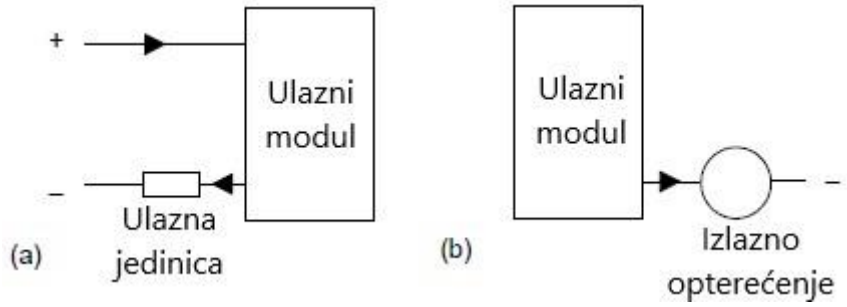
1. Kod relejnog tipa, signal sa izlaza PLC-a se korisiti da bi upravljao relejom i sposoban je uključivati struje red veličina par ampera u eksternom krugu. Osim što omogućuje manjim strujama uključivati veće struje, relej služi da bi izolirao PLC od eksternog kruga. Međutim, releji su relativno spori kod upravljanja. Relejni izlazi su prikladni za paljenje izmjenične i istosmjerne struje. Mogu podnijeti veliki val strujnih i naponskih prijelaznih pojava.
2. Tranzistorski tip izlaza koristi tranzistor za puštanje struje kroz eksterni krug. Ovo daje značajno brži odziv nego kod relejnog tipa. Međutim, ovaj tip se korisiti isključivo kod istosmjerne struje i osjetljiv je na strujna preopterećenja i na visoki

povratni napon. Za zaštitu, koristi se ili osigurač ili ugrađena elektronička zaštita. Optoizolatori se koriste jer osiguravaju izolaciju.

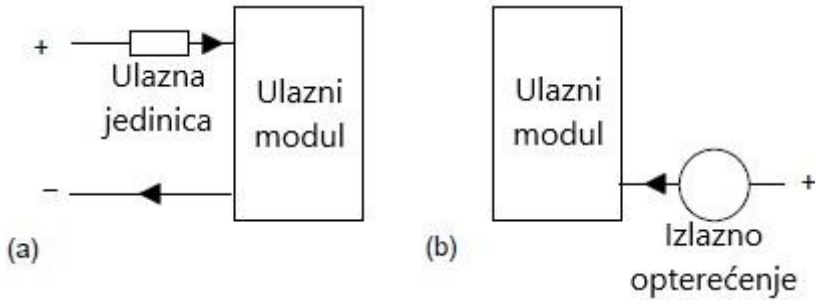
3. Tip trijaka, s optoizolatorima za izolaciju, mogu se koristiti za upravljanje eksternim krugom izmjenične struje. Striktno se koristi za izmjeničnu struju i vrlo lako se uništi strujnim preopterećenjem. Osigurači su gotovo uvijek uključeni da bi zaštitili ovakve izlaze. [4]

3.2.5. Sourcing i sinking

Termini *sourcing* i *sinking* se koriste za opisivanje načina na koji su uređaji istosmjerne struje spojeni s PLC-om. Kod *sourcing-a,* koristeći konvencionalni smjer kretanja struje od pozitivnog prema negativnom, ulazna jedinica prima struju od ulaznog modula, pr. ulazni modul je izvor struje. Ako struja teče od izlaznog modula prema izlaznom opterećenju onda izlazni modul referiramo kao *sourcing*. Kod *sinking-a*, koristeći konvencionalni smjer toka struje od pozitivnog prema negativnom, ulazna jedinica snabdijeva struju ulaznom modulu, pr. ulazni modul je *sink* za struju. Ako struja teče prema izlaznom modulu od izlaznog opterećenja onda je izlazni modul referiran kao *sinking*.



Slika 16. Sourcing [4]



Slika 17. Sinking [4]

3.3. Sustavi PLC-a

Postoje dva učestala tipa mehaničkog dizajna za PLC sustave: *single box*, i modularni tip. *Single box* tip se učestalo koristi kod malih programibilnih kontrolera i isporučuje se kao integralni kompaktni paket upotpunjen s napajanjem, procesorom, memorijom i ulazno/izlaznim jedinicama. Tipično ovakvi PLC-ovi mogu imati 6, 8, 12 ili 24 ulaza i 4, 8 ili 16 izlaza i memoriju koja može spremiti od 300 do 1000 instrukcija. [4]

Slika 18. Siemens SIMATIC S7 - 1200 [7]

Sustavi s velikim brojem ulaza i izlaza će vjerojatno biti modularni i dizajnirani da stanu na vodilice. Modularni tip se sastoji od odvojenih modula za napajanje, procesor, itd., koji su često montirani na vodilice unutar metalnog ormarića. Ugradbeni tip se koristi za sve veličine programibilnih kontrolera i ima razne funkcionalne jedinice upakirane u individualne module koje se mogu jednostavno ugraditi u ormare. Miks modula potrebnih za specifičnu svrhu je odabran od strane korisnika. Prema tome, prilično je jednostavno proširiti broj ulazno/izlaznih priključaka dodavanjem dodatnih ulazno/izlaznih modula ili za proširenje memorije dodavanjem dodatnih memorijskih jedinica. [4]

3.4. Programiranje PLC-a

Naprave za programiranje mogu biti ručne naprave, stolne konzole ili računalo. Tek kad je program upisan na napravu za programiranje i kad je spreman, prebacuje ga se na memorijsku jedinicu PLC-a:

1. Ručne naprave za programiranje normalno imaju dovoljnu količinu memorije da bi zadržale programe dok ih se trasportira s jedne lokacije na drugu.
2. Stolne konzole će vjerojatno imati vizualni display koji sadrži punu tipkovnicu i ekranski prikaz.
3. Osobna računala su konfigurirana kao radne postaje za razvoj programa. Dok je nekim PLC-ovima potrebno samo računalo s odgovarajućim software-om, drugima su potrebne specijalne kartice za komunikaciju s računalom. Velika prednost korištenja računala je ta da program može biti spremljen na tvrdi disk ili CD, i jednostavno pravljenje kopija.

Svaki proizvođač PLC-a ima vlastiti software za programiranje svojih uređaja. Mitsubishi ima MELSOFT, Allen-Bradley ima RSLogix a Siemens ima SIMATIC STEP 7. Siemensov program je potupno usklađen s internacionalnim standardom IEC 61131-3 za PLC programske jezike. Kod STEP 7, programeri mogu odabrati između različitih programskih jezika. Osim ljestvičastog dijagrama (LAD) i funkcionalnog blok dijagrama, STEP 7 Basis također uključuje instrukcijsku listu (STL) programski jezik. Postoje dodatne opcije za IEC 61131-3 programske jezike kao što je strukturirani tekst (ST) imena SIMATIC S7-SCL ili sekvencijalni funkcijski dijagram (SFC) nazvan SIMATIC S7-Graph koji omogućava učinkovit način za grafički opis sekvencijskih kontrolnih sustava. Značajke cjelokupnog sistema uključuju mogućnosti sistemske dijagnostike, procesni dijagnostički alati, PLC simulacije, udaljeno održavanje, i dokumentacija pogona. S7-PLCSIM je mogući opcionalni paket za STEP 7 koji omogućuje simulaciju SIMATIC S7 kontrolnih platformi i testiranje programa na osobnom računalu, omogućujući testiranje i refiniranje prije instalacije fizičkog hardware-a. Ranim testiranjem u razvoju projekta, moguće je unaprijediti ukupnu kvalitetu projekta. Instalacija i puštanje u rad može biti brže i jeftinije jer je moguće rano otkrivanje grešaka u programu koje je tad moguće ispraviti. [4]

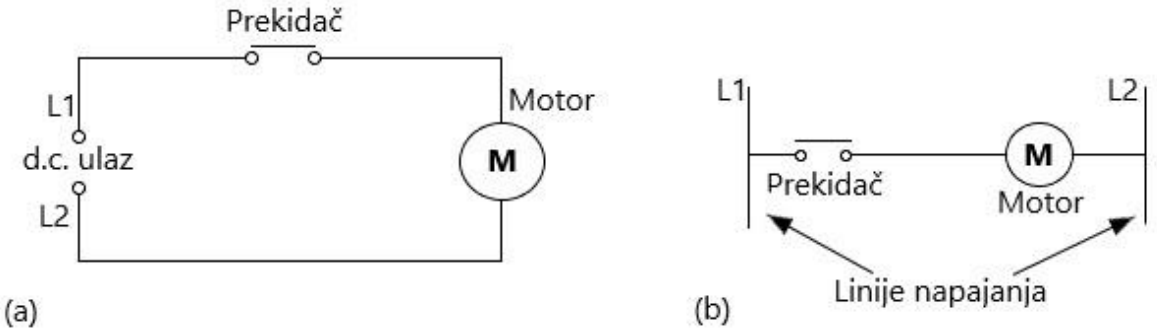
4. Ljestvičasto i funkcionalno blok programiranje

Programi za sustave bazirane na mikroprocesorima moraju biti učitani u njih u strojnom jeziku, a to je sekvenca binarnih brojeva koja predstavlja programske instrukcije. No ipak, asemblerski jezik baziran na uporabi mnemotehike se može koristiti, pr. ljestvičasti dijagram se koristi da bi naznačio operaciju potrebnu za učitavanje podataka koji prate ljestvičasti dijagram, i kompjuterski program nazvan asembler se koristi za prevođenje mnemotehnike u strojni jezik. Programiranje može biti još lakše upotrebom tzv. jezika više razine, pr. BASIC, PASCAL, FORTRAN, COBOL. Ovi jezici koriste unaprijed određene funkcije, predstavljene jednostavnim riječima ili simbolima koji predstavljaju dotičnu funkciju. Primjerice, kod C jezika simbol & se koristi za logičku operaciju I. Međutim, korištenje ovih metoda za pisanje programa zahtjeva određene vještine u programiranju a PLC-ovi su namijenjeni da bi ih koristili inženjeri bez velikog znanja programiranja. Kao posljedica ovoga, *ladder* programiranje je razvijeno. Ovo je način pisanja programa koji onda može biti pretvoren u strojni kod nekim od softwarea da bi ih mogli korisititi PLC mikroprocesori.

Ova metoda pisanja programa je prihvaćena od većine proizvođača PLC-a, no ipak je svaki bio sklon razvoju vlastite verzije pa je zbog toga prihvaćen internacionalni standard za ljestvičasto programiranje kao i za ostale metode programiranja PLC-a. Standard, objavljen 1993. godine, je IEC 1131-3 (*International Electrotechnical Commission*). IEC 1131-3 programski jezici su ljestvičasti dijagrami (LAD), instrukcijska lista (IL), sekvencionalni funkcijski grafovi (SFC), strukturirani tekst (ST), i funkcijski blok dijagrami (FBD). [4]

4.1. Ljestvičasti dijagrami

Kao uvod u ljestvičaste dijagrame, može se uzeti primjer jednostavnog strujnog kruga na Slika 19. (a). Dijagram prikazuje krug za paljenje i gašenje elektromotora. Ovakav dijagram se može nacrtati na drugi način, koristeći dvije vertikalne linije koje predstavljaju ulazne spojnice snage između kojih se spoje ostali elementi kruga. Slika 19. (b) prikazuje rezultat toga. Oba kruga imaju prekidač serijski spojen s motorom koji je spojen na mrežu kad je prekidač zatvoren. Krug prikazan na slici 19. (b) predstavlja ljestvičasti dijagram.



Slika 19. Načini crtanja istog električkog kruga [4]

Kod takvih dijagrama napajanje je uvijek prikazano kao dvije vertikalne linije dok je ostatak kruga prikazano u horizontalnim linijama. Linije napajanja su vertikalne stranice ljestava dok su horizontalne linije prečke na ljestvama. Horizontalne prečke prikazuju samo kontrolni dio kruga, na slici 19. to bi bio samo prekidač u seriji s motorom. Shema strujnog kruga obično prikazuju relativnu lokaciju komponenti kruga i kako su one zapravo spojene. Kod ljestvičastih dijagrama fizička lokacija komponente nije bitna i naglasak je samo na pokazu kako se provodi upravljanje. [4]

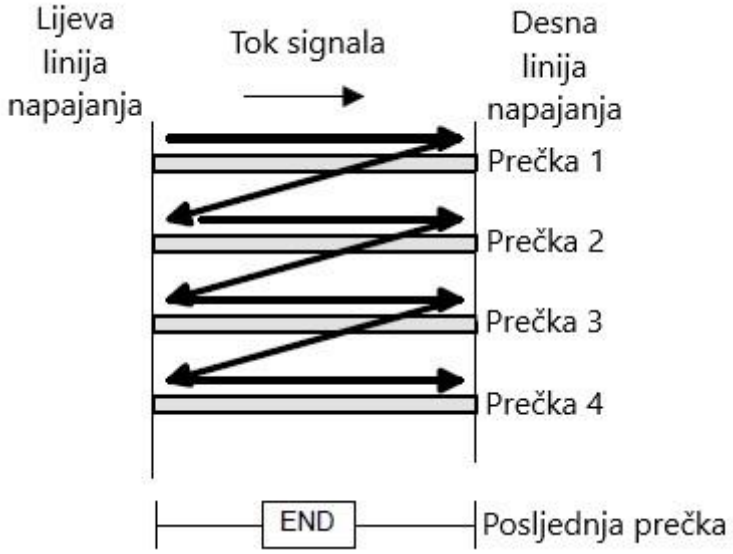
4.1.1. Ljestvičasto programiranje PLC-a

Vrlo česta metoda programiranja PLC-a je bazirana na korištenju ljestvičastih dijagrama. Ljestvičasti dijagram se sastoji od dvije vertikalne linije koje predstavljaju linije napajanja. Krugovi su spojeni kao horizontalne linije, slično prečkama na ljestvama, između tih vertikala.

Prilikom crtanja ljestvičastog dijagrama vrijede niža pravila:

1. Vertikalne linije dijagrama predstavljaju linije napajanja između kojih su krugovi spojeni. Tok struje je s lijeva na desno.
2. Svaka prečka (horizontalna linija) ljestava predstavlja jednu operaciju u kontrolnom procesu.
3. Ljestvičasti dijagram se čita s lijeva na desno i od vrha prema dnu, Slika 20. prikazuje na koji način PLC skenira ljestvičasti dijagram. Prvo se čita najviša linija s lijeva na desno, pa druga linija s lijeva na desno, itd. Dok je PLC u radnom režimu, prođe kroz cijeli ljestvičasti dijagram do kraja, posljednja prečka dijagrama je jasno označena pa odmah kreće od početka. Procedura prolaska kroz sve linije programa se naziva ciklus.

Posljednja linija može biti označena blokom s riječi END ili RET za povratak, jer se program odmah vraća na početak.

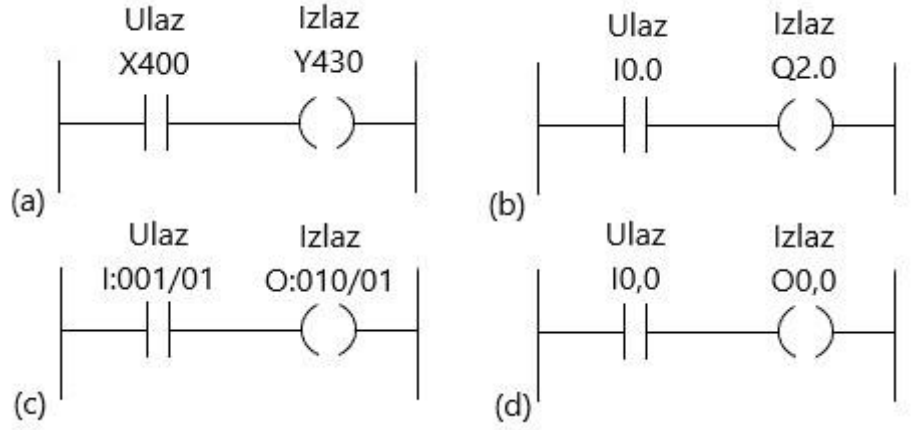


Slika 20. Skeniranje ljestvičastog dijagrama [4]

4. Na početku svake prečke mora biti ulaz ili ulazi a na kraju bar jedan izlaz. Pojam ulaz se koristi kod akcija upravljanja, ako što je zatvaranja kontakta prekidača korištenih kod ulaza u PLC. Pojam izlaz se koristi kod uređaja spojenih na izlaz PLC-a, pr. motor.
5. Električki uređaji su prikazani u njihovom normalnom stanju. Pa tako prekidač koji je normalno otvoren dok ga neki objekt ne zatvori, je prikazan otvorenim u ljestvičastom dijagramu. Prekidač koji je normalno zatvoren je prikazan zatvorenim.
6. Određeni uređaj se može pojaviti u više od jedne horizontalne linije. Npr., u sustavu može biti relej koji pali jedan ili više uređaja. Ista slova i/ili brojevi se koriste za označavanje tog uređaja u svakoj situaciji.
7. Ulazi i izlazi su svi definirani vlastitom adresom, a koristi se notacija definirana od proizvođača PLC-a. Ovo je adresa memorije ulaza ili izlaza PLC.

Kod crtanja ljestvičastih dijagrama imena asocirana s varijablama ili adresa svakog elementa je dodana svom simbolu. Slika 21. prijazuje ljestvičasti dijagram i notaciju adresa kod raznih proizvođača: (a) Mitsubishi, (b) Siemens, (c) Allen-Bradley, (d) Telemecanique. Pa tako slika 20(a) prikazuje da ova prečka ljestvice ima ulaz s adrese X400 i izlaz s adrese

Y430. Prilikom spajanja ulaza i izlaza s PLC-om, relevantni moraju biti spojeni s ulaznim i izlaznim terminalima s tim adresama. [4]



Slika 21. Notacije: (a) Mitsubishi, (b) Siemens, (c) Allen-Bradley, (d) Telemecanique [4]

4.2. Logičke operacije

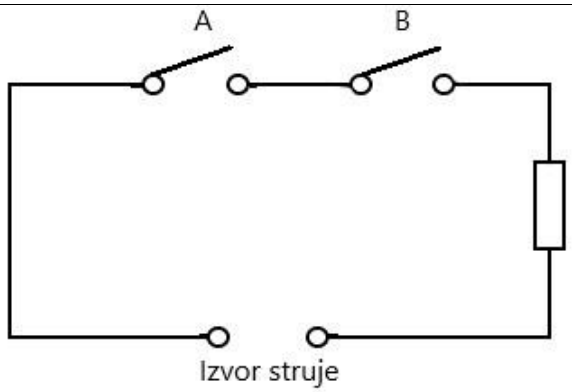
Postoje određene situacije upravljanja kad je potrebna realizacija određenih uvjeta. Niže je opis osnovnih logičkih operacija.

4.2.1. Logička operacija I

Slika 22. prikazuje situaciju gdje izlaz nije aktiviran ako dva, normalno otvorena, prekidača nisu oba zatvorena. Prekidač A i prekidač B moraju biti zatvoreni u isto vrijeme što u biti daje logičku operaciju **I**. Stanje 1 ukazuje na uključeni signal dok stanje 0 ukazuje na isključeni signal, pa ako je cilj da stanje izlaza bude 1 onda prekidači A i B moraju imati stanje 1. Ovakva operacija se kontrolira pomoću logičkih vrata i veza između ulaza do logičkih vrata je upisana u tablicu istine. Tako za **I** vrata vrijedi:

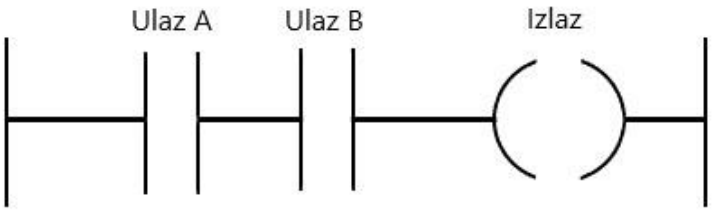
Tablica 1. Tablica istine logičke operiacije I

ULAZ		IZLAZ
A	**B**	
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**



Slika 22. Logičko I u shemi strujnog kruga

Slika 23. prikazuje logiku **I** u ljestvičastom dijagramu. Ulaz A i ulaz B imaju simbol normalno otvorenih ulaza | |. Da bi na izlazu bilo signala, ulazi A i B moraju biti u stanju logičke jedinice odnosno moraju biti zatvoreni.



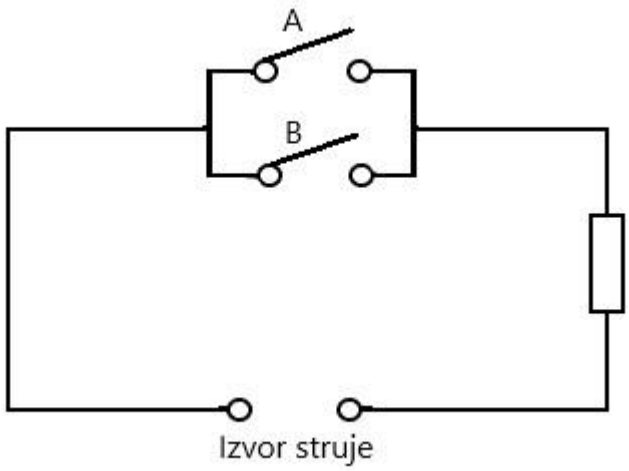
Slika 23. Logičko I u ljestvičastom dijagramu

4.2.2. Logička operacija ILI

Slika 24. prikazuje električki krug gdje je izlaz pod naponom kad prekidači A ili B, oba normalno otvoreni, su zatvoreni. Ovo opisuje logička vrata ILI u tome da ulaz A ili ulaz B moraju biti uključeni kako bi bilo izlaza. Tablica istine je:

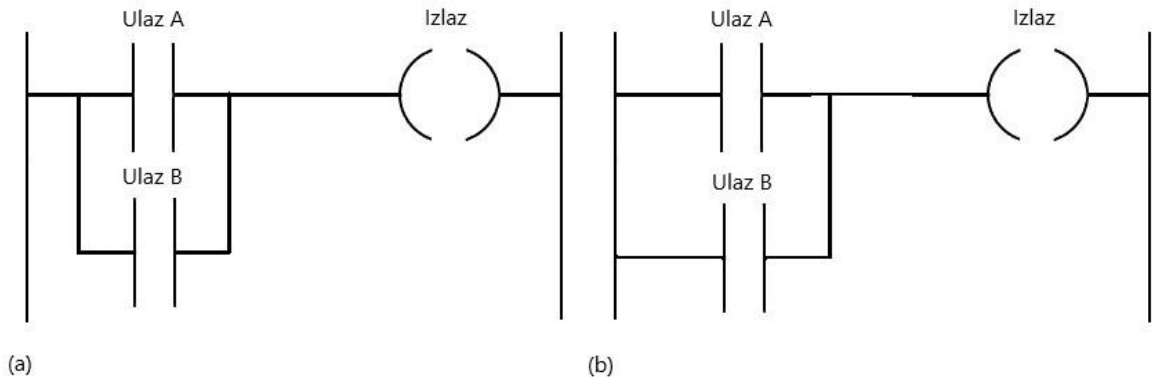
Tablica 2. Tablica istine logičke operacije ILI

ULAZ		**IZLAZ**
A	**B**	
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**



Slika 24. Logičko ILI u shemi strujnog kruga

Slika 25.(a) prikazuje sistem logičkih vrata ILI u ljestvičastom dijagramu, Slika 25.(b) prikazuje ekvivalentni alternativni način crtanja istog dijagrama. Ljestvičasti dijagram starta s | |, normalno otvorenim kontaktima označenima s ulaz A koji predstavlja prekidač A, i u paraleli s njim ulaz B koji predstavlja prekidač B. Ili ulaz A ili ulaz B moraju biti zatvoreni da bi izlaz bio pod naponom.



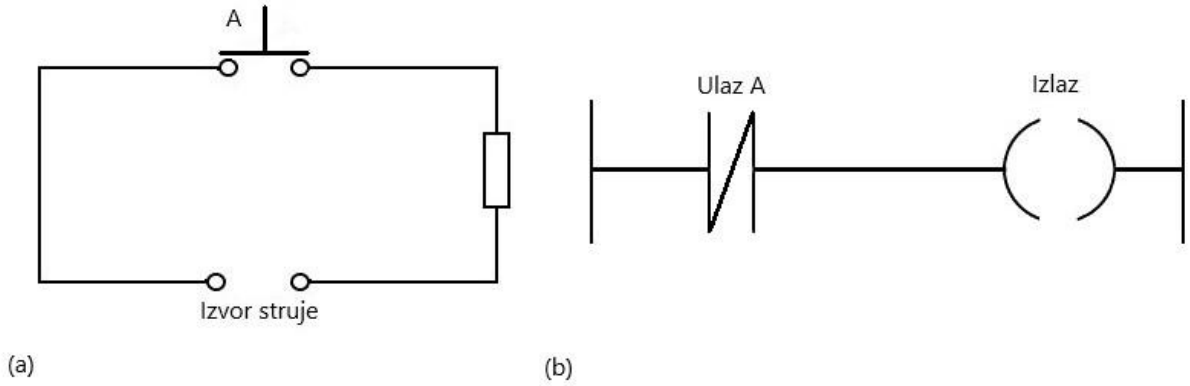
Slika 25. Logička vrata ILI

4.2.3. Logička operacija NE

Slika 26. (a) prikazuje električki krug upravljan prekidačem koji je normalno zatvoren. Kad na prekidaču postoji ulaz, on se otvori i onda nema struje u krugu. Ovo ilustrira logička vrata NE u tome da izlaz postoji kad nema ulaza i da nema izlaza kad postoji ulaz. Ova logička vrata se nekad nazivaju inverter. Tablica istine je:

Tablica 3. Tablica istine logičke operacije NE

ULAZ	IZLAZ
A	
0	**1**
1	**0**



Slika 26 . (a) NE električki krug, (b) NE logika u ljestvičastom dijagramu

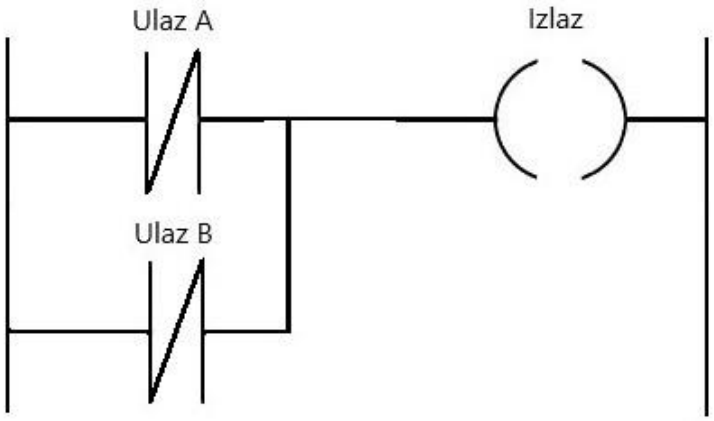
Slika 26. (b) prikazuje sistem logičkih vrata NE u ljestvičastom dijagramu. Kontakti ulaza A su prikazani kao normalno zatvoreni. Ulaz A je u seriji s izlazom ( ). Kad nema ulaza na ulazu A, kontakti su zatvoreni i na izlazu ima signala. Kad na ima signala na ulazu A, on se otvori i tada nema izlaza.

4.2.4. Logička operacija NI

Kod logičkih vrata NI, oba ulaza A i B moraju biti u stanju 0 da bi izlaz imao stanje 1. Izlaz je aktivan kad ulaz A i ulaz B nisu u stanju 1. Kombinacija ovih vrata se nazivaju NI vrata. Tablica istine je:

Tablica 4. Tablica istine logičke operacije NI

ULAZ		IZLAZ
A	**B**	
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**



Slika 27. Logičko NI u ljestvičastom dijagramu

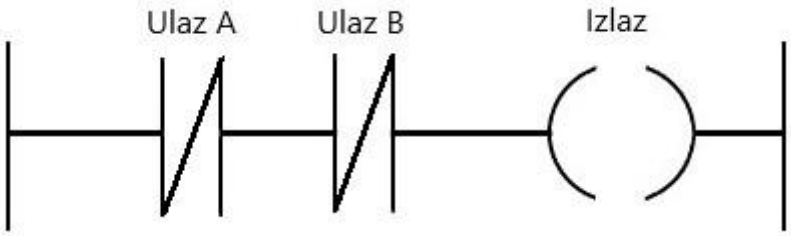
4.2.5. Logička operacija NILI

Ako bi nakon ILI vrata slijedila vrata NE, za posljedicu bi svaki izlaz s ILI vrata bio invertiran. Alternativa bi bila na svaki ulaz postaviti NE vrata i onda zajednička I vrata, pa bi rezultat bio isti. Niže je tablica istine:

Tablica 5. Tablica istine logičke operacije NILI

ULAZ		IZLAZ
A	**B**	
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**0**

Kombinacija ILI i NE vrata se naziva NILI vrata. Izlaz postoji kad ni jedan od ulaza nema stanje 1. Slika 28. prikazuje ljestvičasti dijagram NILI sustava.



Slika 28. Logičko NILI u ljestvičastom dijagramu

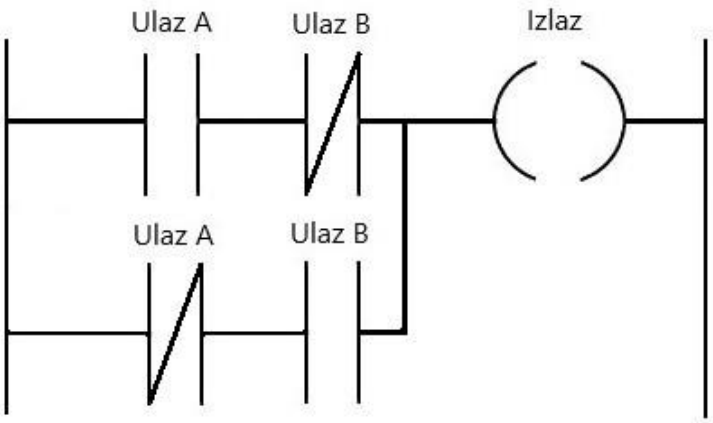
4.2.6. Logička operacija Ekskluzivno ILI

ILI vrata daju izlaz kad ili jedan ili oba izlaza imaju stanje 1. Ponekad postoji potreba za logičkim vratima kod kojih je izlaz aktivan ako bilo koji od ulaza ima stanje 1, ali ne i ako oba istovremeno imaju stanje 1. Niže je tablica istine:

Tablica 6. Tablica istine logičke operacije Ekskluzivno ILI

ULAZ		IZLAZ
A	**B**	
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Takva vrata se nazivaju Ekskluzivna ILI ili XILI vrata. Slika 29. prikazuje ljestvičasti dijagram za logička vrata XILI. Kad ulaz A i ulaz B nisu aktivirani onda izlaz ima stanje 0. Kad je aktiviran ulaz A, onda gornja grana rezultira stanjem 1 kod izlaza. Kad je ulaz B aktiviran, onda je izlaz aktiviran preko donje grane. Kad su oba ulaza aktivirana onda nema signala na izlazu.

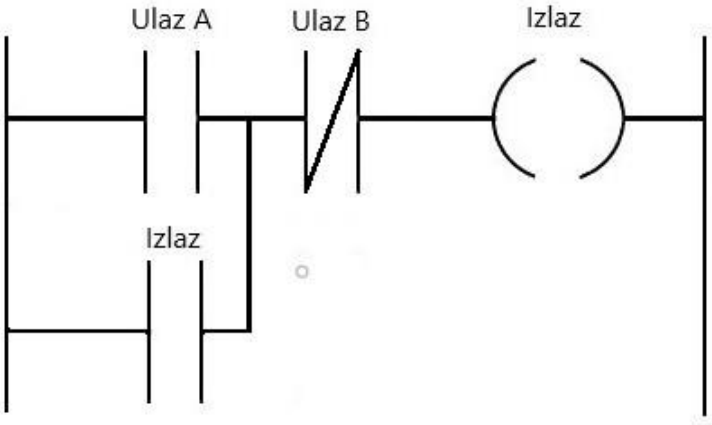


Slika 29. Ekskluzivno ILI u ljestvičastom dijagramu

4.3. Fiksiranje

Često postoje situacije kad je potrebno držati izlaz pod naponom tj. u stanju 1, čak i kad signal sa ulaza nestane. Jednostavan primjer takve situacije je kad je motor uključen pomoću dugmeta. Iako kontakti u ovom slučaju ne ostaju zatvoreni, postoji zahtjev da motor radi dok se ne aktivira dugme STOP. Pojam *latch circuit* se koristi kod krugova koji izvršavaju ovu radnju. Ovo je samoodrživi krug jer nakon što se dovede napon, on zadržava svoje stanje dok ne dobije signal s drugog ulaza.

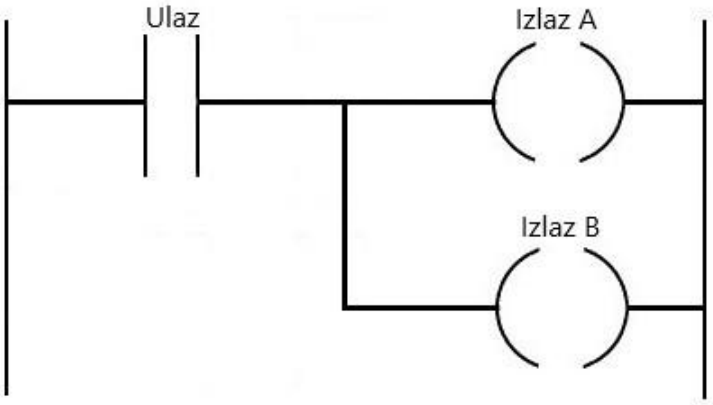
Primjer fiksiranog kruga je prikazan na Slika 30. Kad se kontakti na ulazu A zatvore, izlaz je u stanju 1. Međutim, kad je izlaz u stanju 1, drugi set kontakata asociran s izlazom se zatvori. Ovi kontakti čine logička vrata ILI s ulaznim kontaktima. Prema tome, ako se kontakti ulaza A otvore, krug će i dalje držati izlaz pod naponom. Jedini način za otpuštanje kontakta izlaza je aktivirajući normalno zatvoreni kontakt B.



Slika 30. Fiksirani krug

4.4. Višestruki izlazi

Kod ljestvičastih dijagrama može postojati više od jednog izlaza spojenog na kontakt. Slika 31. prikazuje ljestvičasti dijagram s dva izlaza. Kad se kontakti ulaza zatvore oba izlaza prelaze u stanje 1.



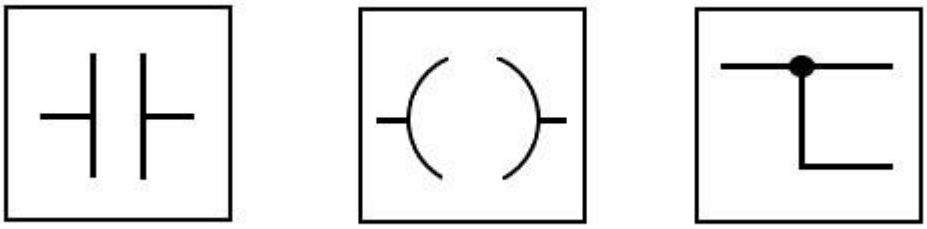
Slika 31. Ljestvičasti dijagram s dva izlaza

4.5. Unošenje programa

Svaka horizontalna prečka na ljestvicama predstavlja instrukciju u programu koju će PLC koristiti. Cijele ljestve daju kompletan program. Postoji više raznih metoda za upis programa u terminal za programiranje. Koja god metoda je korištena za unos programa u terminal za programiranje ili računalo, izlaz do memorije PLC-a mora biti u formi koju može koristiti mikroprocesor. Ovo se naziva strojni jezik i radi se o binarnom kodu, pr. 0010100001110001.

4.5.1. Ljestvičasta metoda programiranja

Jedna od metoda unošenja programa u terminal za programiranje uključuje uporabu tipkovnice koja ima tipke sa simbolima koji predstavljaju razne elemente ljestvičastog dijagrama i unose ih tako da se ljestvičasti dijagram prikazuje na ekranu terminala za programiranje. Slika 32. Prikazuje neke od simbola ljestvičastog dijagrama. S lijeva na desno su simboli za ulaz, izlaz i početak raskrsnice.



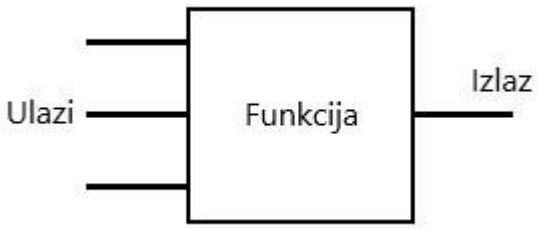
Slika 32. Neki od simbola ljestvičastog dijagrama

Terminal onda prevede program nacrtan na ekranu u strojni jezik.

Računala se mogu koristiti za crtanje ljestvičastog programa. Za ovo je potrebno instalirati relevantan software, pr. RSLogix od Rockwell Automation Inc. za Allen-Bradley PLC-ove, MELSOFT – GX Developer za Mitsubishi PLC-ove, STEP 7 – Micro/WIN za Siemens PLC-ove, itd. Software radi na operacijskom sustavu Windows a simboli se postavljaju *drag and drop* metodom.

4.6. Funkcionalni blokovi

Pojam funkcionalni blok dijagram (FBD) se koristi kod PLC programa opisanih pomoću grafičkih blokova. Opisan je kao grafički jezik za prikaz signala i toka podataka kroz blokove, koji su u suštini elementi software-a. Funkcijski blok je programska instrukcija koja , kad je izvršena, daje jednu ili više vrijednosti izlaza. Blok je prikazan na način kao na Slika 33. gdje se unutar bloka unosi ime funkcije.



Slika 33. Funkcijski blok

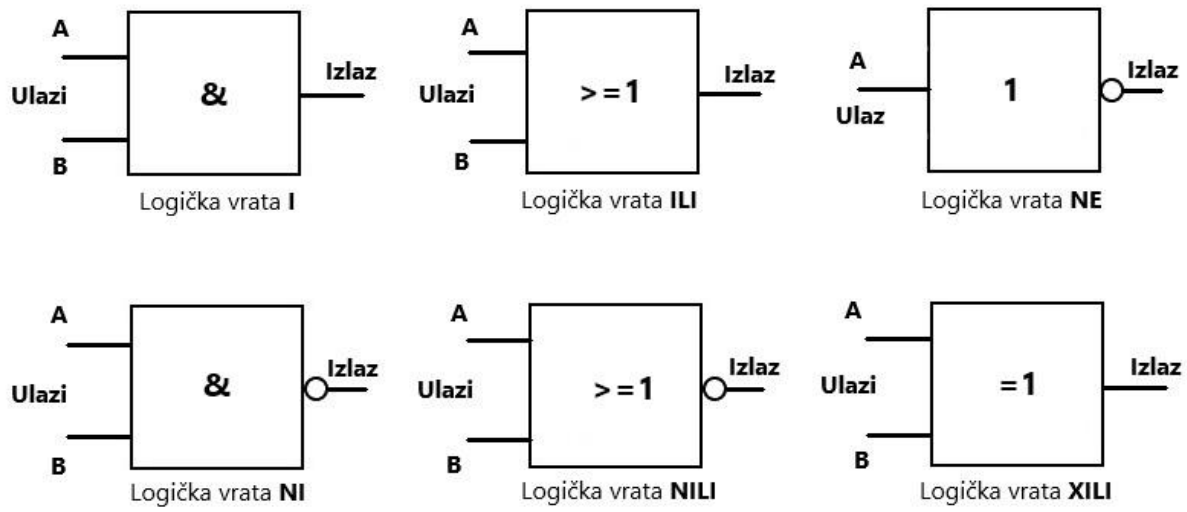
Prema standardu IEC 113-3, funkcionalni blok je prikazan kao pravokutan blok s ulazima na lijevoj strani i izlazima na desnoj strani. Unutar bloka je ime funkcije koju

obavlja, a ime bloka je iznad njega. Imena ulaza u blok su smještena na prikladna mjesta, unutar bloka uz točku najbliže tom ulazu.

Funkcijski blokovi mogu imati standardne funkcije , kao što su one od logičkih vrata, brojača i timera, ili imati funkcije određene od korisnika, pr. blok za dobiti srednju vrijednost ulaza.

4.6.1. Logička vrata

Programi se često sastoje od logičkih vrata. Postoje dva standarda simbola koji se koriste za logička vrata, jedan tip koji je nastao u SAD-u gdje svaki simbol ima svoj oblik, a drugi je internacionalni standard (IEEE/ANSI) koji koristi pravokutnik s logičkom funkcijom upisanom unutar njega. 1 u pravokutniku ukazuje na to da izlaz postoji dok je ulaz 1. ILI funkcija se notira kao ≥ 1, a ovo je zato jer izlaz postoji kad je ulaz veći ili jedak 1. Negirani ulaz je prikazan kao mali krug na ulazu, a negativan izlaz kao mali krug na izlazu. Slika 34. prikazuje funkcionalne blok dijagrame u IEEE/ANSI formi.



Slika 34. Simboli logičkih vrata [4]

4.6.2. Booleova algebra

Ljestvičasti programi se mogu izvesti iz Boolovih izraza jer je matematički sustav logike od interesa. U Booleovoj algebri postoje samo dvije znamenke, 0 i 1. Kod **I** operacije za ulaze A i B se može pisati:

$$A * B = Q$$

gdje Q predstavlja izlaz. Prema tome Q je jednak 1 samo kad su A = 1 i B = 1.

ILI operacija s ulazima A i B se piše:

$A + B = Q$

Q je jednak 1 samo kad je A = 1 ili B = 1.

NE operacija s ulazom A se piše:

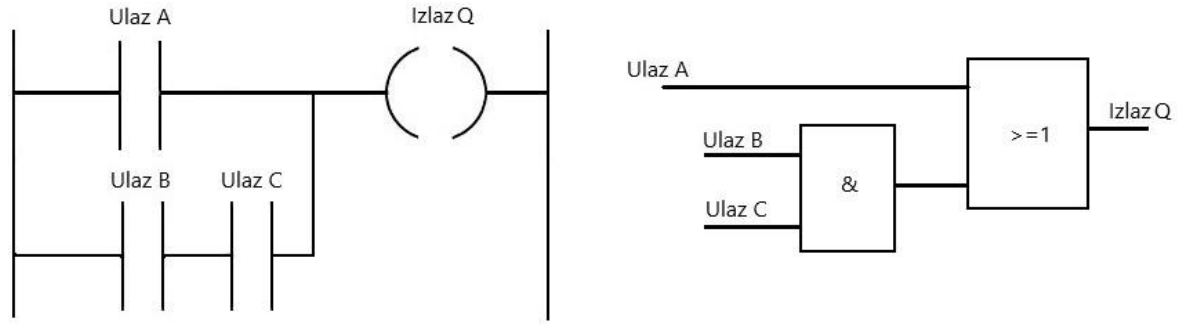
$\bar{A} = Q$

Kad A nije 1 izlaz je prisutan.

Kao primjer kako povezati Booleove izraze s ljestvičastim dijagramima za primjer je uzet izraz: $A + B * C = Q$

Ovo govori da ako je A jednako 1 ili B i C istovremeno u stanju 1 onda postoji izlaz Q. Slika 35. prikazuje ljestvičasti dijagram i funkcionalni blokovski dijagram. Prema Siemensovoj notaciji ovo bi izgledalo:

$I0.0 + I0.1 * I0.2 = Q2.0$



Slika 35. Ljestvičasti dijagram A + B * C = Q

4.7. Ostale metode programiranja

Ovdje će se kratko spomenuti ostale metode programiranja prema IEC 1131-3. To su instrukcijske liste (*instruction lists* – IL), sekvencionalni funkcijski dijagrami (*sequential function charts* – SFC), i strukturirani tekst (*structured text* – ST).

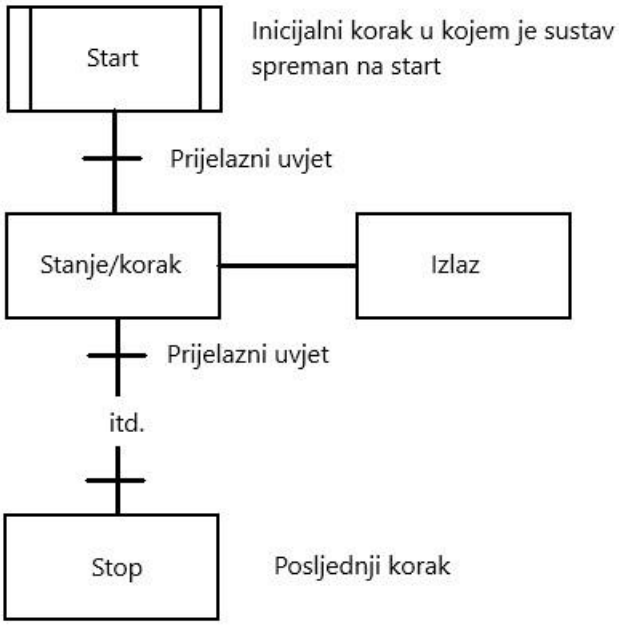
4.7.1. Instrukcijske liste

Programska metoda koja bi predstavljala unošenje ljestvičastog programa koristeći tekst se zove instrukcijske liste. Instrukcijske liste daju programe koji se sastoje od niza instrukcija, svaka instrukcija na novoj liniji. Instrukcija se sastoji od operatora koje slijede jedan ili više operanada, pr. subjekata operatora. Kod ljestvičastih dijagrama operatorom se može smatrati ljestvičasti element. Svaka instrukcija može koristiti ili izmjeniti vrijednost spremljenu u memorijski registar. Za ovo se koriste mnemonički kodovi gdje svaki kod odgovara određenom operatoru/ljestvičastom elementu. Kodovi se razlikuju od proizvođača do proizvođača, iako je preporučeni standard IEC 1131-3 uglavnom prihvaćen.

4.7.2. Sekvencionalni funkcijski dijagrami

Pojam sekvencionalni funkcijski dijagram se koristi za slikovnu reprezentaciju rada sustava da bi prikazali sekvence događaja uključenih u rad sustava. Sastoje se od sljedećih elemenata:

1. Operacija je opisana brojem odvojenih sekvencionalno spojenih stanja ili koraka koji su prikazani kao pravokutnici, gdje svaki predstavlja određeno stanje sustava koji se kontrolira. Inicijalni korak u programu je predstavljen drugačije od drugih koraka, Slika 36. pokazuje njegovu reprezentaciju.



Slika 36. Stanje i njegova tranzicija [4]

2. Svaka spojna linija između stanja ima horizontalnu crtu koja predstavlja prijelazni uvjet koji mora biti realiziran prije nego sustav prijeđe iz jednog stanja u drugo. Dva koraka nikad ne mogu biti direktno povezana, uvijek moraju biti odvojeni prijelazom.. Dva prijelaza ne mogu nikad direktno pratiti jedan drugog, uvijek moraju biti odvojena korakom.
3. Kad su prijelazni uvjeti do idućeg stanja realizirani onda se iduće stanje ili korak u sustavu dogodi.
4. Proces nastavlja od jednog stanja do drugog sve dok se ne izvrši kompletni ciklus.
5. Izlazi/akcije u bilo kojem stanju su predstavljene horizontalno povezanim pravokutnicima i dogode se kad je povezano stanje realizirano.

4.7.3. Strukturirani tekst

Strukturirani tekst je programski jezik koji jako nalikuje programskom jeziku PASCAL. Programi se pišu kao skup izjava odvojenih točka-zarezom. Izjave koriste predefinirane izjave i podrutine za promjenu varijabli, a varijable su određene vrijednosti, interno spremljene vrijednosti ili ulazi i izlazi.

Strukturirani tekst nije osjetljiv na velika i mala slova pa ih se može koristiti prema potrebi i da bi se pojednostavnilo shvaćanje koda. Isto tako razmak nije potrebno koristiti ali može se također radi pojednostavljenja razumijevanja, to isto vrijedi za uvučene linije. [4]

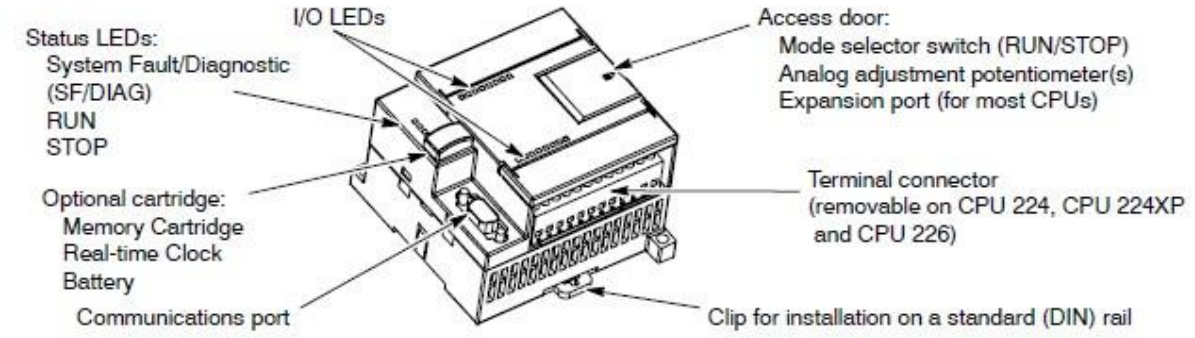
5. Siemens SIMATIC S7-200

Serija S7-200 je linija mikro PLC-ova koji mogu upravljati raznim aplikacijama u automatici. Kompaktan dizajn, niska cijena, moćni set instrukcija čine S7-200 jako dobrim rješenjem za upravljanje manjim aplikacijama. Uz širok raspon modela i programski alat prilagođen Windowsu, S7-200 daje fleksibilnost koja je potrebna za rješavanje problema automatizacije.

S7-200 motri ulaze i mijenja izlaze prema korisničkom programu koji može uključivati Booleovu logiku, brojanje, mjerenje vremena, kompleksne matematičke operacije, komunikaciju s drugim inteligentnim uređajima. [8]

S7-200 se sastoji od mikroprocesora, integriranog napajanja, ulaznih krugova, izlaznih krugova u kompaktnom kučištu. Nakon šte se program učita u kontroler, S7 – 200 sadrži logiku potrebnu za praćenje i kontrolu ulaznih i izlaznih jedinica.

5.1. S7-200 CPU



Slika 37. S7-200 Micro PLC [8]

Siemens snabdjeva različite S7-200 procesore s različitim značajkama i mogućnostima koje pomažu kod nalaska efektivnih rješenja za različite aplikacije.

Ovaj kontroler ima 4 modela procesora CPU 221, CPU 222, CPU 224XP, i CPU 226. Razlikuju se po dimenzijama kučišta, količini programske memorije, memorije podataka,

backupa memorije u slučaju nestanka struje, broju ulaza/izlaza, broju modula koje je moguće priključiti, itd... Slika 38. prikazuje CPU224XP.

Slika 38. S7-200 CPU 224XP [9]

Tablica 7. Usporedba S7-200 CPU modela [8]

Feature	CPU 221	CPU 222	CPU 224	CPU 224XP	CPU 226
Physical size (mm)	90 x 80 x 62	90 x 80 x 62	120.5 x 80 x 62	140 x 80 x 62	190 x 80 x 62
Program memory: with run mode edit without run mode edit	 4096 bytes 4096 bytes	 4096 bytes 4096 bytes	 8192 bytes 12288 bytes	 12288 bytes 16384 bytes	 16384 bytes 24576 bytes
Data memory	2048 bytes	2048 bytes	8192 bytes	10240 bytes	10240 bytes
Memory backup	50 hours typical	50 hours typical	100 hours typical	100 hours typical	100 hours typical
Local on-board I/O Digital Analog	 6 In/4 Out –	 8 In/6 Out –	 14 In/10 Out –	 14 In/10 Out 2 In/1 Out	 24 In/16 Out –
Expansion modules	0 modules	2 modules[1]	7 modules[1]	7 modules[1]	7 modules[1]
High-speed counters Single phase Two phase	 4 at 30 kHz 2 at 20 kHz	 4 at 30 kHz 2 at 20 kHz	 6 at 30 kHz 4 at 20 kHz	 4 at 30 kHz 2 at 200 kHz 3 at 20 kHz 1 at 100 kHz	 6 at 30 kHz 4 at 20 kHz
Pulse outputs (DC)	2 at 20 kHz	2 at 20 kHz	2 at 20 kHz	2 at 100 kHz	2 at 20 kHz
Analog adjustments	1	1	2	2	2
Real-time clock	Cartridge	Cartridge	Built-in	Built-in	Built-in
Communications ports	1 RS-485	1 RS-485	1 RS-485	2 RS-485	2 RS-485
Floating-point math	Yes				
Digital I/O image size	256 (128 in, 128 out)				
Boolean execution speed	0.22 microseconds/instruction				

5.2. Ekspanzijski moduli

Da bi PLC bio što fleksibilniji za rješavanje raznih zadataka, serija S7-200 uključuje širok raspon ekspanzijskih modula. Ekspanzijski moduli služe da bi dodali dodatnu funkcionalnost osnovnoj jedinici. Tablica 8. prikazuje osnovne značajke modula za proširenje. [8]

Tablica 8. S7-200 ekspanzijski moduli [10]

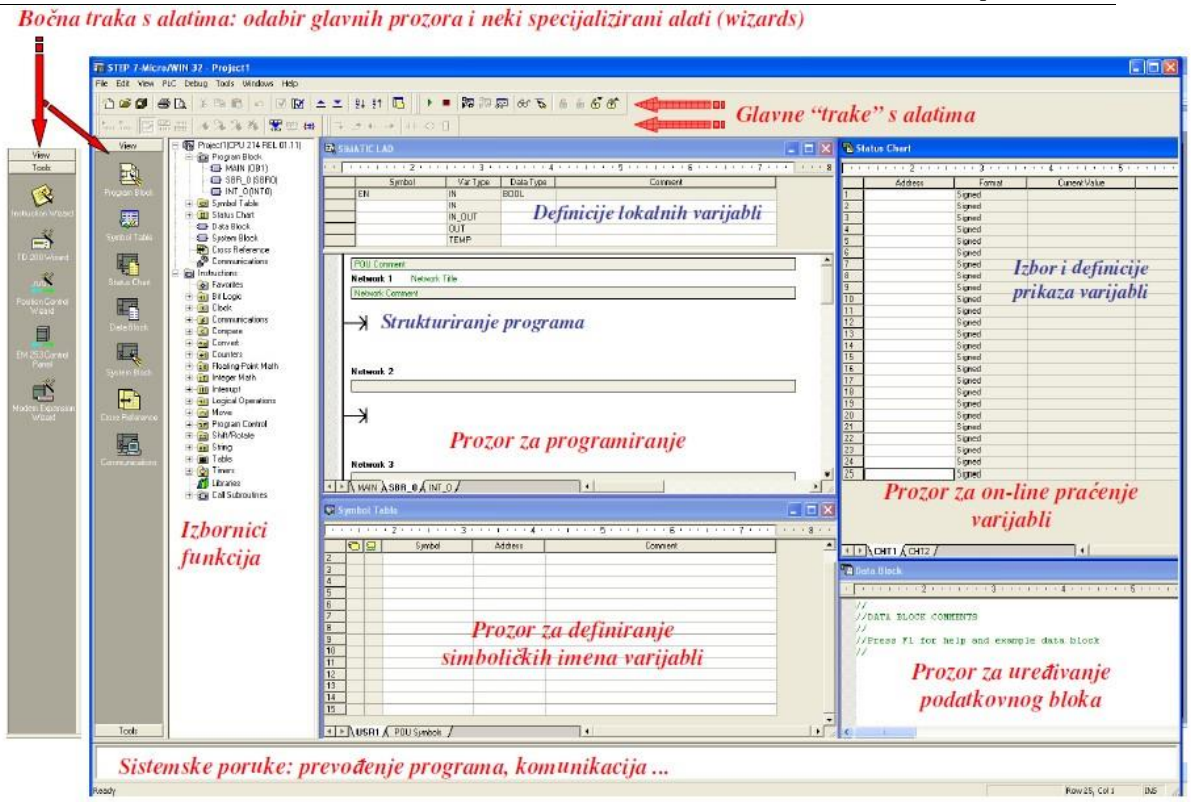
Expansion Modules		Types		
Discrete modules	Input	8 x DC In	8 x AC In	16 x DC In
	Output	4 x DC 8 x DC Out	4 x Relays 8 x AC Out	 8 x Relay
	Combination	4 x DC In / 4 x DC Out 4 x DC In / 4 x Relay	8 x DC In / 8 x DC Out 8 x DC In / 8 x Relay	16 x DC In/16 x DC Out 16 x DC In/16 x Relay
Analog modules	Input	4 x Analog In	4 x Thermocouple In	2 x RTD In
	Output	2 x Analog Out		
	Combination	4 x Analog In / 1 Analog Out		
Intelligent modules		Position Ethernet	Modem Internet	PROFIBUS-DP
Other modules		AS-Interface		

Slika 39. Tipične komponente SIMATIC S7-200 sustava [10]

5.3. STEP 7-Micro/WIN programski paket

STEP 7-Micro/WIN programski paket je ugodan za korištenje, odnosno pruža jednostavnost razvoja, izmjene, i praćenja logike potrebne za upravljanje procesom. Postoje 3 programska editora za praktičnost i učinkovitost u razvoju upravljačkog programa za odabranu aplikaciju. Isto tako, ukoliko je potrebna dodatna pomoć odnosno informacije, sve je dostupno online ili u priručniku za S7-200.

STEP 7-Micro/WIN se može pokrenuti na osobnom računalu ili na Siemensovom uređaju za programiranje, kao što je PG 760. Minimalni zahtjev za pokretnje programskog paketa su Windowsi 2000 ili Windows XP , 100 Mb slobodnog prostora na hard disku i miš jer se radi na *drag and drop* način. Slika 40. prikazuje prozor u programskom paketu STEP 7-Micro/WIN.



Slika 40. STEP 7-Micro/WIN [8]

5.4. Komunikacijske opcije

Siemens pruža dvije opcije programiranja za spajanje računala s S7-200. direktna konekcija s PPI Multi-Master kabelom, ili kartica komunikacijskog procesora (*Communications Processor* – CP) s MPI kabelom.

PPI Multi-Master programski kabel je najčešća i ekonomski najprikladnija metoda spajanja računala sa S7-200. Ovaj kabel spaja komunikacijski priključak od PLC-a sa serijskom komunikacijom računala. Ovaj kabel se može koristiti za spajanje drugih komunikacijskih naprava sa PLC-om. [8]

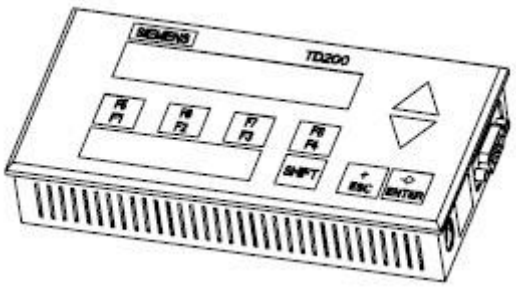
Slika 41. Multi-master kabel [10]

5.5. Vrste zaslona

Na S7-200 je moguće montirati više od jednog tipa panela za upravljanje i praćenje rada sustava. Ukoliko postoje prostorna ograničenja koristi se tekstualni display koji može prikazati od 2 do 4 linije i takvi su prikladni za jednostavnije procese. A kod složenijih procesa koriste se grafički paneli koji se upravljaju na dodir, i koji mogu pokazati više stavki procesa od jednom.

5.5.1. Tekstualni display (TD 200 i TD 200C)

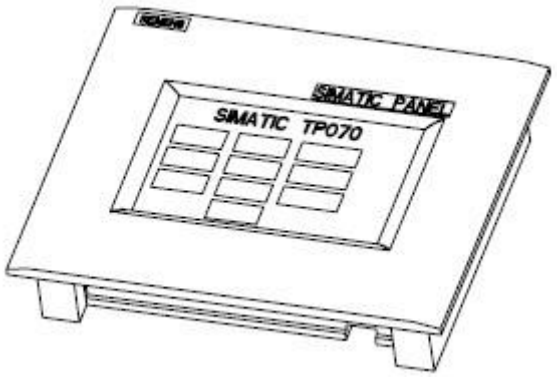
TD 200 i TD 200C imaju zaslon u 2 linije, 20 znakova po liniji. Ovo je dovoljno da PLC prikaže tekstualne poruke i ostale vrste podataka koji se tiču upravljanog procesa. Radi se o zaslonima niske cijene koji dozvoljavaju praćenje i promjene procesnih varijabli koje se tiču procesa kojim se upravlja. [8]



Slika 42. Zaslon TD 200 i TD 200C [8]

5.5.2. TP070 i TP170 zasloni

TP070 i TP170 su zasloni upravljani na dodir koje je moguće spojiti s S7-200. Ovi zasloni pružaju mogućnost uređenja zaslona po želji, mogu prikazivati željenu grafiku, varijable procesa, grafove i slično, a sve preko user-friendly zaslona na dodir.



Slika 43. Zaslon upravljan na dodir [8]

5.6. Princip rada S7-200

S7-200 konstantno ciklira kroz kontrolnu logiku u programu, čitajući i pišući podatke. Osnovne operacije ovog PLC-a su veoma jednostavne:

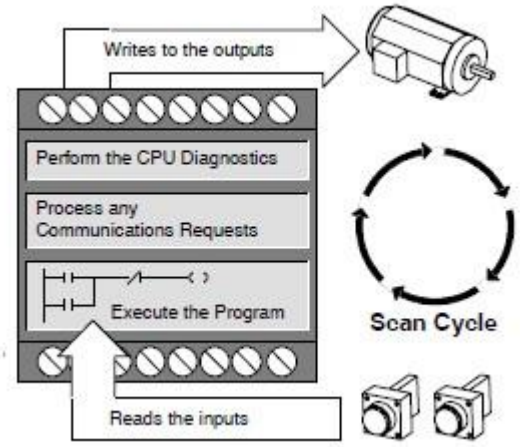
- S7-200 čita statuse ulaza.
- Program spremljen u S7-200 koristi ove ulaze za procjenu upravljačke logike. Dok program teče, S7-200 nadopunjuje podatke.
- S7-200 upisuje ove podatke na izlaze.

PLC izvodi serije zadataka ponavljajući. Ova ciklička egzekucija zadataka se naziva ciklus skeniranja. Kako je prikazano na Slika 44. PLC izvodi većinu ili sve nadolazeće zadatke tijekom ciklusa skeniranja:

- Dok čita ulaze: S7-200 kopira stanja fizičkih ulaza u registre stanja logičkih ulaza.
- Izvršavanje kontrolne logike u programu: S7-200 izvršava instrukcije programa i sprema vrijednosti u različita memorijska područja.
- Procesiranje svih komunikacijskih zahtjeva: S7-200 izvršava sve zadatke potrebne za komunikaciju.
- Izvršavanje CPU samo-testne dijagnostike: S7-200 osigurava da *firmware*, programska memorija, i bilo koji ekspanzijski moduli rade ispravno.

- Upisivanje na izlaze: vrijednosti spremljene u registre stanja logičkih izlaza su upisane na fizičke izlaze.

Egzekucija korisničkog programa ovisi o tome da li je S7-200 u STOP modu ili u RUN modu. U RUN modu, program se izvršava, dok u STOP mogu, program se ne izvršava. [8]



Slika 44. S7-200 ciklus skeniranja [8]

5.6.1. Čitanje ulaza

Digitalni ulazi: Svaki ciklus skeniranja počinje s čitanjem trenutne vrijednosti digitalnih ulaza i onda pisanje tih vrijednosti u registar stanja logičkih ulaza.

Analogni ulazi: S7-200 ne ažurira analogne ulaze s ekspanzijskih modula kao dio normalnog ciklusa skeniranja osim ako filtriranje analognih ulaza nije omogućeno. Analogni filter služi tomu da bi mogli imati stabilniji signal. Analogni filter se može omogućiti za svaku analognu ulaznu točku.

Kad je omogućeno filtriranje analognog ulaza, S7-200 ažurira taj analogni ulaz jednom po ciklusu skeniranja, izvodi funkciju filtriranja, i sprema filtrirane vrijednosti interno. Svaki put kad program pristupa analognom ulazu isporučuje mu se filtrirana vrijednost.

Kad analogno filtriranje nije omogućeno, S7-200 čita vrijednosti analognog ulaza iz ekspanzijskih modula svaki put kad program pristupi analognom ulazu. [8]

5.6.2. Izvršavanje programa

Tijekom egzekucijske faze ciklusa skeniranja, S7-200 izvršava dani program, startajući od prve instrukcije i nastavlja sve do posljednje instrukcije. Neposredne I/O instrukcije daju neposredni pristup ulazima i izlazima tijekom egzekucije bilo programa bilo prekidne rutine.

Ako se koriste prekidi u programu, prekidne rutine asocirane s prekidnim događajima se spremaju kao dio programa. Prekidne rutine se ne izvršavaju kao dio normalnog ciklusa skeniranja, ali se izvršavaju kad se prekidni događaj pojavi (a to može biti u bilo kojem trenutku ciklusa skeniranja).

Tijekom faze procesiranja poruka u ciklusu skeniranja, S7-200 procesira sve poruke koje su primljene od komunikacijskog porta ili inteligentnih I/O modula.

Na kraju svakog ciklusa skeniranja, S7-200 upisuje vrijednosti spremljene u registar stanja logičkih izlaza u digitalne izlaze. (Analogni izlazi su ažurirani neposredno, nezavisno od ciklusa skeniranja.) [8]

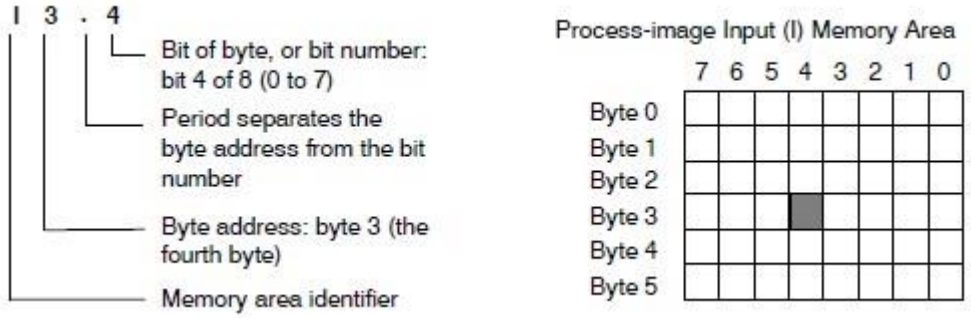
5.7. Pristup podatcima od S7-200

S7-200 sprema informacije u različite memorijske lokacije koje imaju jedinstvene adrese. Moguće je eksplicitno navesti memorijsku adresu kojoj želimo pristupiti. Ovo omogućava danom programu direktni pristup informacijama. Tablica 9. prikazuje raspon cjelobrojnih vrijednosti koji mogu biti prikazani različitim veličinama podataka.

Tablica 9. Decimalni i heksadecimalni rasponi za različite veličine podataka [8]

Representation	Byte (B)	Word (W)	Double Word (D)
Unsigned Integer	0 to 255 0 to FF	0 to 65,535 0 to FFFF	0 to 4,294,967,295 0 to FFFF FFFF
Signed Integer	-128 to +127 80 to 7F	-32,768 to +32,767 8000 to 7FFF	-2,147,483,648 to +2,147,483,647 8000 0000 to 7FFF FFFF
Real IEEE 32-bit Floating Point	*Not applicable*	*Not applicable*	+1.175495E-38 to +3.402823E+38 (positive) -1.175495E-38 to -3.402823E+38 (negative)

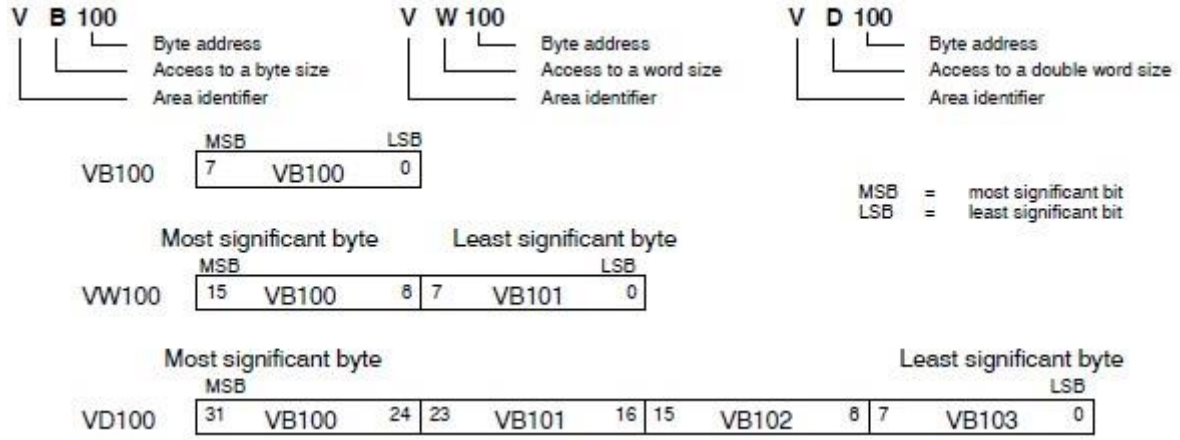
Za pristup bitu u memorijskom području, potrebno je specificirati adresu, koja uključuje identifikator memorijskog mjesta, adresu bajta, i broj bita. Slika 45. prikazuje primjer pristupa bitu (koji se također naziva 'byte.bit' adresiranje). U ovom primjeru, memorijsko područje i adresa bajta (I = *input* = ulaz, i 3 = bajt 3) su slijeđeni točkom ('.') da bi razdvojila adresu bita (bit 4).



Slika 45. Pristup podacima unutar 'byte' podatka [10]

Moguće je pristupiti podatcima u većini memorijskih područja (V, I, Q, M, S, L, i SM) kao bitovi, riječi, dvostruke riječi (double words – 32 bitovni format) koristeći bajt-adresa format. Za pristup bajtu, riječi ili dvostrukoj riječi od podataka iz memorije, potrebno je specificirati adresu na način sličan kao kod specificiranja adrese bita. Ovo uključuje identifikator područja, vrste podatka po veličini, i početne bajt adrese od bajta, riječi i dvostruke riječi kao što je prikazano na Slika 46.

Podacima na ostalim memorijskim područjima (kao što su T, C, HC, i akumulatori) se pristupa koristeći format adrese koji uključuje identifikator područja i broj uređaja. [8]



Slika 46. Pristup podacima duljine 8, 16 i 32 bita [10]

Kod pristupa podatcima duljine *BYTE*, *WORD* i *DWORD* važno je obratiti pažnju gdje se nalaze MSB i LSB. MSB se nalazi na najnižoj adresi i on definira početnu *BYTE* adresu podatka. [10]

5.8. Pristup podacima na memorijskim mjestima

5.8.1. Registar stanja logičkih ulaza: I

S7-200 uzorkuje fizičke ulaze na početku svakog ciklusa skeniranja i upisuje ove vrijednosti u registre stanja logičkih ulaza. Registru stanja logičkih ulaza se može pristupiti u bitovima, bajtovima, riječima ili dvostrukim riječima:

Bit: **I*[byte adresa].[bit adresa]***

I0.1

Byte, Word, DWORD: **I*[veličina][adresa početnog byte]***

IB4

5.8.2. Registar stanja logičkih izlaza: Q

Na kraju svakog ciklusa skeniranja, S7-200 kopira vrijednosti spremljene iz registra stanja logičkih izlaza na fizičke izlaze. Process-image ulaznom registru se može pristupiti u bitovima, bajtovima, riječima ili duplim riječima:

Bit: **Q*[byte adresa].[bit adresa]***

Q1.1

Byte, Word, DWORD: **Q*[veličina][adresa početnog byte]***

QB5

5.8.3. Područje varijabilne memorije: V

V memorija se može koristiti za spremanje posrednih rezultata operacija koje se izvode pomoću kontrolne logike u programu. V memorija se isto tako može koristiti za spremanje drugih podataka koji se tiču upravljanog procesa ili zadatka. V memoriji se može pristupiti u bitovima, bajtovima, riječima ili dvostrukim riječima:

Bit: **V*[byte adresa].[bit adresa]***

V10.2

Byte, Word, DWORD: **V*[veličina][adresa početnog byte]***

VW100

5.8.4. Područje bit memorije: M

Područje M memorije se može koristiti kao kontrolne releje za spremanje posrednih statusa operacija ili drugih kontrolnih informacija. Području M memorije se može pristupiti u bitovima, bajtovima, riječima ili dvostrukim riječima:

Bit: **M*[byte adresa].[bit adresa]***

M26.7

Byte, Word, DWORD: **M*[veličina][adresa početnog byte]***

MD20

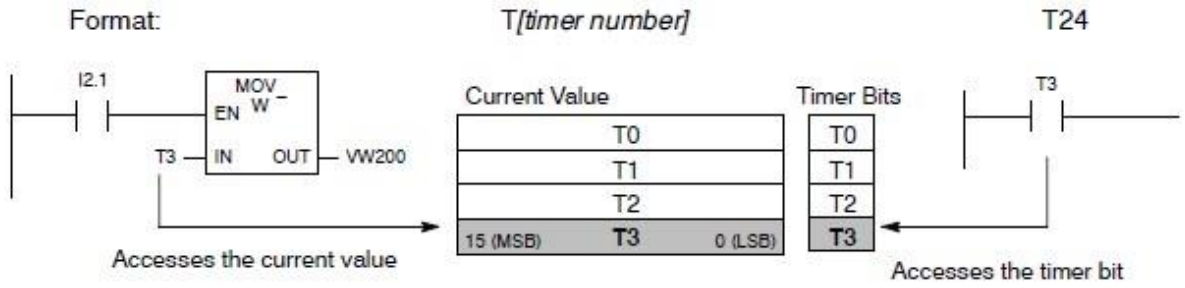
5.8.5. Područje timer memorije: T

S7-200 ima timere koji broje inkremente vremena u rezolucijama od 1 ms, 10 ms, ili 100 ms. Dvije varijable su povezane s timerom:

- Trenutna vrijednost: ovaj 16-bitni argument sprema količinu vremena izbrojanog od timera.

- Timer bit: ovaj bit je postavljen ili izbrisan kao rezultat usporedbe trenutne i postavljene vrijednosti. Postavljena vrijednost je unesena kao dio instrukcije timeru.

Ovim varijablama se može pristupiti koristeći adresu timera (T + broj timera). Pristup bilo timer bitu bilo trenutnoj vrijednosti je ovisno o korištenoj instrukciji: instrukcije s bit operandima pristupaju timer bitu, dok instrukcije s word operandima pristupaju trenutnoj vrijednosti. Kako je prikazano na Slika 47. instrukcija s normalno otvorenim kontaktom pristupa timer bitu, dok Move Word instrukcija pristupa trenutnoj vrijednosti timera. [8]



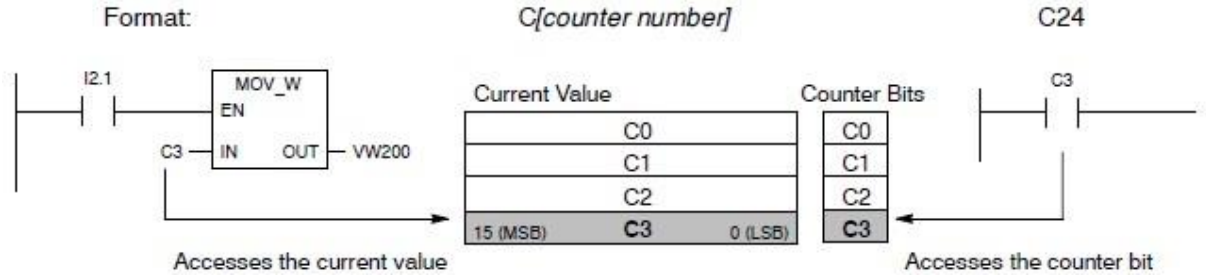
Slika 47. Pristupanje timer bitu ili trenutnoj vrijednosti timera [10]

5.8.6. Područje memorije brojila: C

Kod S7-200 postoje tri tipa brojila: jedan tip broji samo prema gore, jedan tip broji samo prema dolje, i jedan tip broji prema gore i dolje. Dvije varijable su povezane s brojilom:

- Trenutna vrijednost: ovaj 16-bitni argument sprema akumulirani broj.
- Bit brojila: ovaj bit je postavljen ili izbrisan kao rezultat usporedbe trenutne i postavljene vrijednosti. Postavljena vrijednost je unesena kao dio instrukcije brojilu.

Ovim varijablama se može pristupiti koristeći adresu brojila (C + broj countera). Pristup bilo counter bitu bilo trenutnoj vrijednosti je ovisno o korištenoj instrukciji: instrukcije s bit operandima pristupaju counter bitu, dok instrukcije s word operandima pristupaju trenutnoj vrijednosti. Kako je prikazano na Slika 48. instrukcija s normalno otvorenim kontaktom pristupa counter bitu, dok Move Word instrukcija pristupa trenutnoj vrijednosti brojila. [8]



Slika 48. Pristupanje counter bitu ili trenutnoj vrijednosti countera [10]

5.8.7. High-Speed counteri: HC

High-speed counteri broje događaje velike brzine neovisno o skeniranju CPU-a. Ovi counteri rade s 32-bitnim podatcima. Da bi pristupili vrijednosti brojila, potrebno je specificirati adresu brojila, koristeći memorijski tip (HC) i broj countera (kao što je HC0). Trenutna vrijednost brojila je read-only vrijednost i moguće ju je adresirati samo kao DWORD (32 bita).

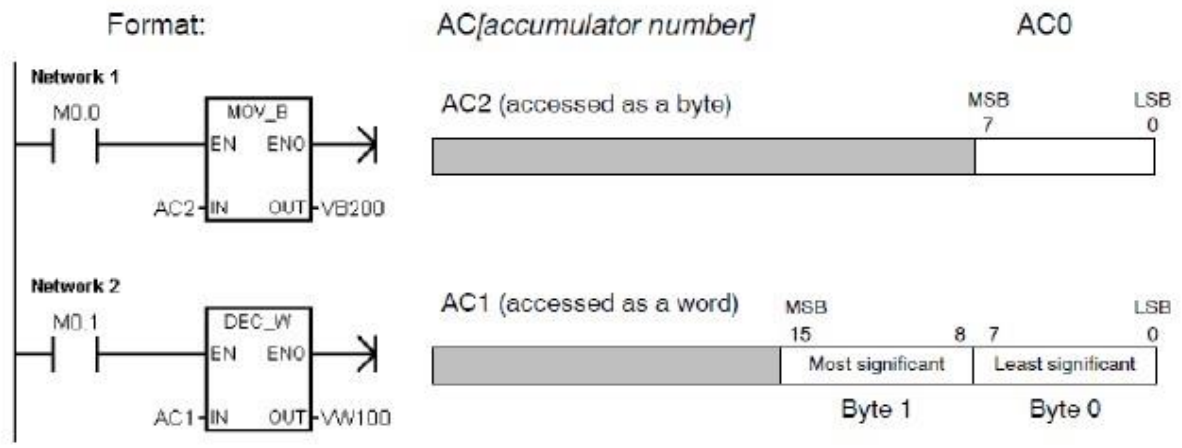
Format: **HC*[broj high-speed countera]***

HC1

5.8.8. Akumulatori: AC

Akumulatori su čitaj/piši uređaji koje je moguće koristiti kao memoriju. Primjerice, akumulatore je moguće koristiti za prosljeđivanje parametara do i od subrutina i za spremanje neposrednih vrijednosti korištenih u kalkulaciji. S7-200 osigurava četiri 32-bitna akumulatora (AC0, AC1, AC2, i AC3). Podatcima iz akumulatora je moguće pristupiti kao bit, byte ili DWORD.

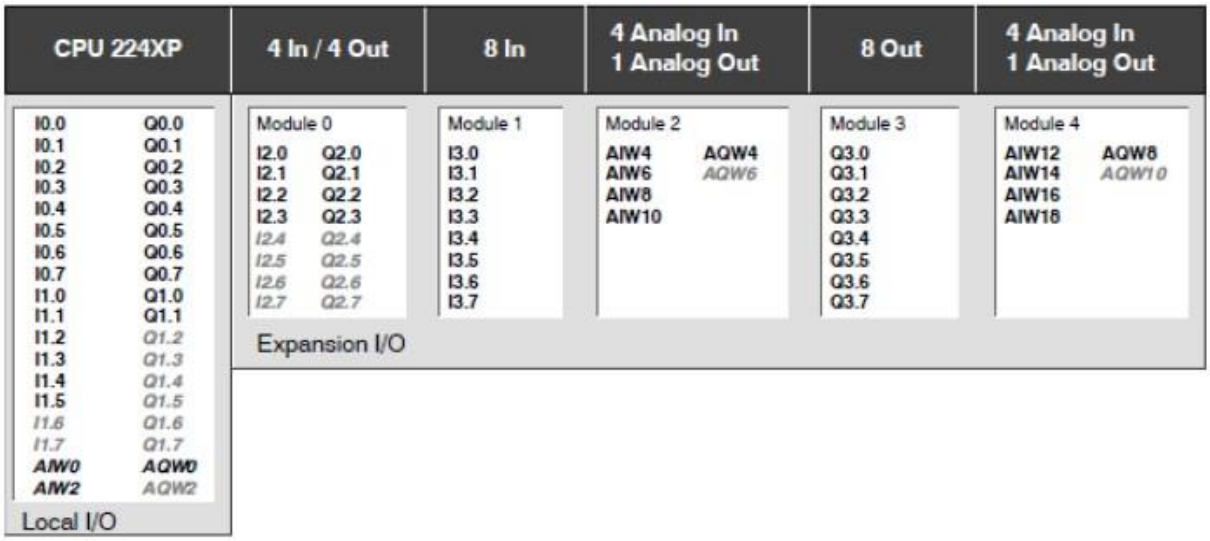
Veličina podatka kojem se pristupa je određena instrukcijom koja se koristi prilikom pristupa akumulatoru. Na Slika 49. je prikazano da za pristup podatcima u akumulatoru (radnom registru) se pristupa ovisno o duljini podatka s kojim akumulator treba obaviti operaciju. Za pristup akumulatoru kao DWORD, koriste se svih 32 bita. [10]



Slika 49. Pristup akumulatorima [10]

5.9. Adresiranje lokalnih i ekspanzijskih I/O

Lokalni I/O na PLC-u imaju postavljene fiksne I/O adrese. Moguće je PLC nadograditi dodatnim ulazima i izlazima spajanjem ekspanzijskih I/O modula s desne strane CPU-a, formirajući I/O lanac. Slika 50. pruža primjer I/O označavanja za specifičnu konfiguraciju hardware-a. "Praznine" u adresiranju prikazane kao sivi italic font ne mogu biti korištene u danom programu. [8]



Slika 50. Primjer I/O adresa za lokalne i ekspanzijske I/O (CPU 224XP) [10]

5.10. Spremanje i vraćanje podataka

S7-200 omogućava različite načine za zadržavanje korisničkog programa i ostalih podataka u memoriji.

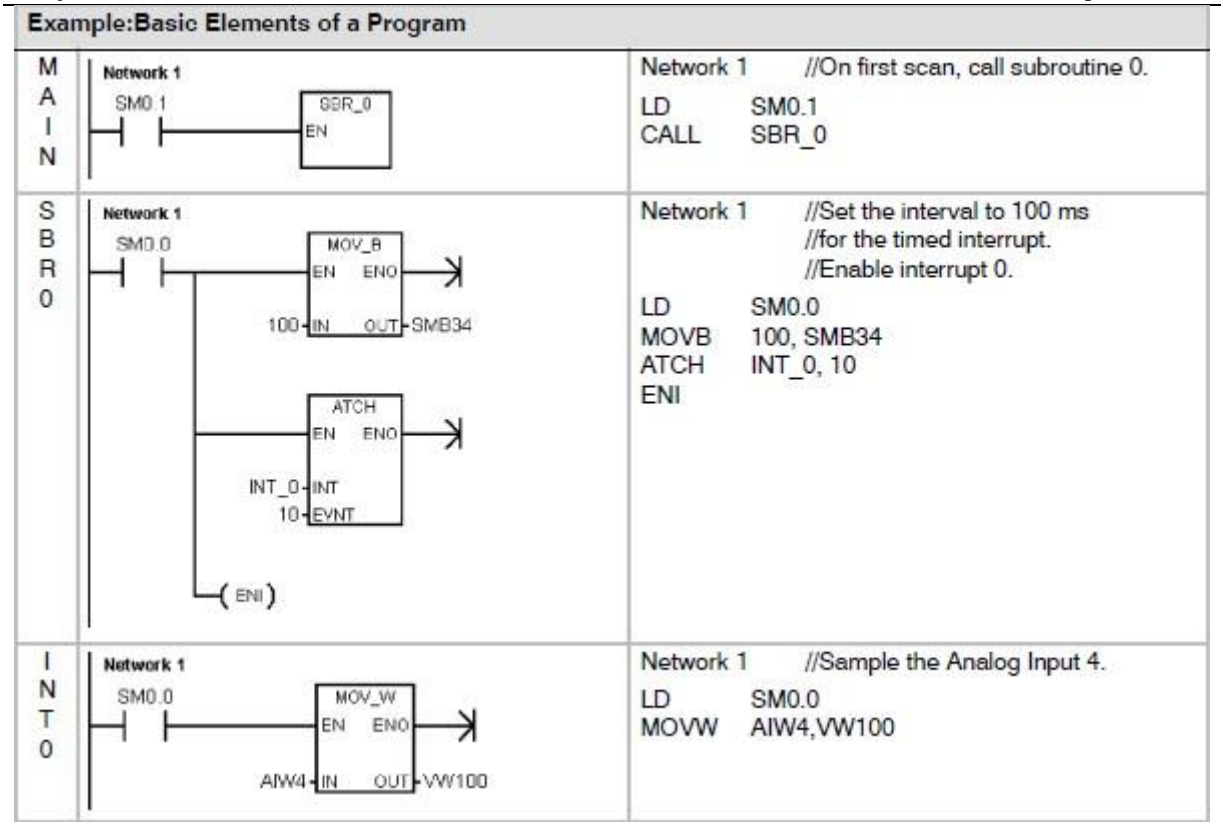
- Retentivna memorija podataka – područja podataka koje korisnik odabire da ostanu nepromjenjeni tijekom ciklusa napajanja, sve dok superkondenzator i opcionalna baterija nije ispražnjene. V, M, trenutno stanje timer i brojila su jedina područja memorijskih podataka koje je moguće konfigurirati da budu retentivna.
- Permanentna memorija – neizbrisiva memorija korištena za spremanje programskih blokova, blokova podataka, sistemskih blokova, forsiranih vrijednosti, M memorije kofigurirane da bi se spremila uslijed gubitka struje, i odabrane vrijednosti upisane od korisnika.
- Spremnik za memoriju – uklonjiva neizbrisiva memorija korištena za spremanje programskih blokova, blokova podataka, sistemskih blokova, zapisnika podataka, i forsiranih vrijednosti

Može se koristiti S7-200 Explorer za spremanje dokumentacijskih fileova (doc, text, pdf, itd.) u spremnik za memoriju. [8]

5.11. Osnovni elementi programa

Programski blok je sastavljen od izvršnog koda i komentara. Izvršni kod se sastoji od glavnog programa i bilo kakvih potprograma ili prekidnih potprograma. Kod je kompajliran i prebačen na S7-200 a komentari programa nisu. Mogu se koristiti organizacijski elementi (glavni program, potprogram, i prekidni potprogrami) za strukturiranje kontrolnog programa.

Slika 51. prikazuje program koji se sastoji od potprograma i prekidnog potprograma. Ovaj primjer koristi vremenski prekid za čitanje vrijednosti analognog ulaza svakih 100 ms.



Slika 51. Primjer programa s osnovnim elementima programa

5.11.1. Glavni program

Glavni dio programa sadrži instrukcije koje upravljaju danom aplikacijom. S7-200 izvršava ove instrukcije sekvencionalno, jednom po ciklusu skeniranja. Glavni program se često referira kako OB1.

5.11.2. Potprogrami

Ovo su opcionalni elementi programa koji se pokreću samo kad su pozvani: od glavnog programa, od prekidnog potprograma, ili od nekog drugog potprograma. Potprogrami su korisni u slučajevima kad je potrebno više puta izvršiti funkciju. Umjesto pisanja logike na svako mjesto u glavnom programu gdje je potrebno izvršavanje te funkcije, tu logiku je moguće napisati u potprogram i pozivati ga koliko god puta je potrebno. Potprogrami pružaju nekoliko benefita:

- Korištenjem potprograma se reducira ukupna veličina programa.

- Korištenjem potprograma se smanjuje vrijeme skeniranja zato jer je kod premješten iz glavnog programa. S7-200 evaluira kod u glavnom programu svaki ciklus skeniranja, bilo da je kod izvršen ili ne, ali S7-200 evaluira kod u potprogramu samo kad se pozove potprogram.
- Korištenje potprograma stvara kod koji je prijenosan. Moguće je izolirati kod da bi funkcionirao kao potprogram i onda ga kopirati u druge programe uz malo ili nimalo prerade. [8]

5.11.3. Prekidni potprogrami

Ovo su opcionalni elementi programa koji reagiraju na specifične prekidne događaje. Prekidni potprogram je dizajniran da bi upravljao predefiniranim prekidnim događajem. Kad god se pojavi specifični događaj, S7-200 pokreće prekidni potprogram. Korištenje prekidnih (*interrupt*) rutina omogućuje brzo reagiranje na važne vanjske događaje, te neke specifične događaje unutar CPU (npr. izvršavanje PID regulatora s fiksnim intervalom uzorkovanja). [10]

Glavni program nije taj koji poziva prekidni potprogram. Prekidni potprogram je asociran za prekidni događaj, i S7-200 izvršava instrukcije u prekidnom potprogramu samo kad se pojavi prekidni događaj. [8]

Kako se pozivi za prekid programa ne mogu predvidjeti, poželjno je optimirati prekidnu rutinu (kako bi se čim prije nastavilo izvršavanje ostatka koda). Također treba paziti na to koje podatke prekidna rutina mijenja (a koji su kasnije dostupni ostatku koda). [10]

5.12. PID algoritam u PLC-u

Kod operacija u stacionarnom stanju, PID regulator regulira izlaznu veličinu da bi pogrešku e sveo na nulu. Greška je razlika između reference i procesne varijable. Princip PID regulatora je baziran na sljedećoj jednadžbi koja izražava izlaz M(t) kao funkciju proporcionalno-integracijsko-derivacijskog djelovanja.

U ovom poglavlju će se koristiti notacija koja se koristi u STEP7-Micro/WIN.

$$M(t) = K_C * e + K_c \int_0^t e\, dt + M_{initial} + K_C * \frac{de}{dt} \quad (1)$$

Gdje su:

$M(t)$ – izlaz petlje u funkciji vremena

K_C – konstanta pojačanja petlje

e – stacionarna pogreška

$M_{initial}$ – inicijalna vrijednost izlaza petlje

Da bi implementirali funkciju upravljanja u digitalno računalo, kontinuirana funkcija mora biti kvantificirana u periodičke uzorke vrijednosti pogreške s narednom kalkulacijom izlaza. Odgovarajuća jednadžba koja je osnova za rješenja preko digitalnog računala je:

$$M_n = K_c * e_n + K_I * \sum_1^n e_x + M_{initial} + K_D * (e_n - e_{n-1}) \quad (2)$$

Gdje su:

M_n – izračunata vrijednost sa izlaza petlje u vremenu n

K_C – konstanta pojačanja petlje

e_n – vrijednost pogreške petlje u vremenu n

e_{n-1} – prethodna vrijednost pogreške petlje (u vremenu e_{n-1})

e_x – vrijednost pogreške petlje u vremenu x

K_I – pojačanje integralnog člana

$M_{initial}$ – inicijalna vrijednost izlaza petlje

K_D – pojačanje derivacijskog člana

Integralni član u (2) je funkcija svih grešaka petlje, od prvog do trenutnog uzorka. Derivacijski član je funkcija trenutnog uzorka i prethodnog uzorka, dok je proporcionalni član samo funkcija trenutnog uzorka. Kod digitalnih računala, nije praktično spremati sve uzorke greške sustava, niti je potrebno.

Zato što digitalno računalo mora izračunati vrijednost izlaza svaki put kad je greška uzorkovana startajući s prvim uzorkom, potrebno je samo spremiti prethodnu vrijednost

greške i prethodnu vrijednost integralnog člana. Moguće je napraviti pojednostavljenu jednadžbu izraza (2) kako bi se mogla riješiti u svakom vremenu uzorkovanja:

$$M_n = K_c * e_n + K_I * e_n + MX + K_D * (e_n - e_{n-1}) \quad (3)$$

Gdje je:

MX – prethodna vrijednost integralnog člana (u vremenu uzorkovanja n – 1)

S7-200 koristi pojednostavljenu i modificiranu jednadžbu (3) dok izračunava izlaznu vrijednost petlje:

$$M_n = MP_n + MI_n + MD_n \quad (4)$$

Gdje su:

M_n – izračunata vrijednost izlaza petlje u vremenu uzorkovanja n

MP_n – vrijednost proporcionalnog člana izlaza petlje u vremenu uzorkovanja n

MI_n – vrijednost integracijskog člana izlaza petlje u vremenu uzorkovanja n

MD_n – vrijednost dervacijskog člana izlaza petlje u vremenu uzorkovanja n

Algoritam regulatora (3) predstavlja vremenski diskretni PID regulator. [10]

5.12.1. Proporcionalni član PID regulatora

Proporcionalni član MP je produkt pojačanja (K_c), koji upravlja osjetljivošću kalkulacije izlaza, i greške (e), koju čine razlika između reference (SP) i varijable procesa (PV) u danom vremenu uzorkovanja. Jednadžba za proporcionalni član u S7-200:

$$MP_n = K_c * (SP_n - PV_n) \quad (5)$$

Gdje su:

MP_n – vrijednost proporconalnog člana u vremenu uzorkovanja n

SP_n – vrijednost reference u vremenu uzorkovanja n

PV_n – vrijednost procesne veličine u vremenu uzorkovanja n

5.12.2. Integracijski član PID regulatora

Integracijski član $\boldsymbol{MI}$ je proporcionalan sumi pogreške kroz vrijeme. Jednadžba za integracijski član u S7-200:

$$\boldsymbol{MI_n = \frac{K_c * T_S}{T_I * (SP_n - PV_n)} + MX} \tag{6}$$

Gdje su:

$\boldsymbol{MI_n}$ – vrijednost integracijskog člana u vremenu uzorkovanja n

$\boldsymbol{T_S}$ – vrijeme uzorkovanja

$\boldsymbol{T_I}$ – integracijska vremenska konstanta

$\boldsymbol{MX}$ – suma svih prethodnih vrijednosti integracijskog člana

Integracijska suma ($\boldsymbol{MX}$) je suma svih prethodnih vrijednosti integracijskog člana. Nakon svake kalkulacije $\boldsymbol{MI_n}$ integracijska suma je ažurirana s vrijednošću $\boldsymbol{MI_n}$. Inicijalna vrijednost integracijske sume je tipično postavljena na izlaznu vrijednost ($\boldsymbol{M_{initial}}$) koja prethodi prvoj kalkulaciji izlazne vrijednosti petlje.

5.12.3. Derivacijski član PID regulatora

Derivacijski član $\boldsymbol{MD}$ je proporcionalan promjeni pogreške. Jednadžba za derivacijski član u S7-200:

$$\boldsymbol{MD_n = \frac{-K_c * T_D}{T_S * (PV_n - PV_{n-1})}} \tag{7}$$

Gdje su:

$\boldsymbol{MD_n}$ – vrijednost derivacijskog člana u vremenu uzorkovanja n

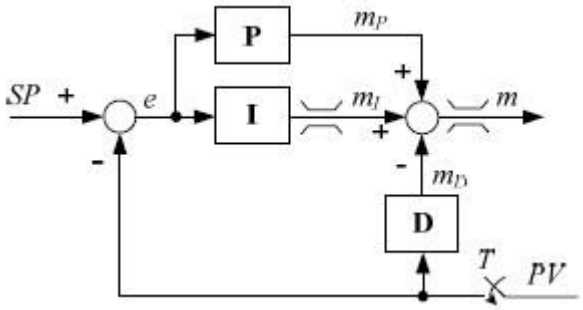
$\boldsymbol{T_D}$ – derivacijska vremenska konstanta

$\boldsymbol{PV_{n-1}}$ – vrijednost procesne veličine u vremenu uzorkovanja n-1

Ovdje se vidi da derivacijski član u obzir uzima promjenu u procesnoj varijabli a ne promjenu u pogrešci. Zbog toga se procesna varijabla mora spremiti jer će se koristiti u

sljedećoj kalkulaciji derivacijskog člana. U vremenu prvog uzorkovanja, vrijednost $\boldsymbol{PV_{n-1}}$ je izjednačena s $\boldsymbol{PV_n}$. [8]

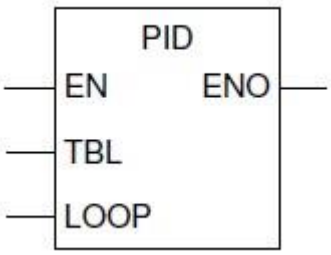
Iz (7) se vidi da derivacijski član ne djeluje na pogrešku sustava nego samo na procesnu veličinu odnosno izlaz sustava. Prema tome, ovdje se radi o modificiranoj izvedbi PID regulatora, takozvani PI + D regulator, prikazan na Slika 52.



Slika 52. PI+D regulator [10]

Također je realizirano i zasićenje integratora (tzv. "reset-antiwindup") kada proračunata vrijednost izlaza regulatora izlazi izvan definiranih graničnih (limit) vrijednosti. Tada se izlaz regulatora limitira, a integralni član se resetira na vrijednost kod koje je suma $m_I(k) - m_P(k) + m_D(k)$ jednaka izlaznom limitu. [10]

Niže je prikazan simbol iz ladder dijagrama za PID regulator:



Slika 53. LAD simbol za PID regulator [10]

TBL – početna adresa polja (tablice) u kojem se nalaze konfiguracijski podatci za PID regulator, podatak tipa byte (VB).
LOOP – broj regulacijske petlje (0 – 7), podatak tipa byte.

Međutim, ukoliko se PID naredba poziva izravno u kodu javlja se problem resetiranja integratora (tzv. PID control manual mode). Zato se preferira korištenje PID Wizard-a za konfiguriranje PID regulatora. Potrebno je skalirati ulaze regulatora, odnosno referencu (setpoint, SP) i izlaz procesa (process value, PV) na raspon 0 ... 1. Shodno tome, izlaz regulatora (M_n) baš kao i stanje integratora (MX) također mogu poprimiti vrijednosti samo u rasponu 0 ... 1.

6. Uvod u sintezu PID regulatora za PTn model

Mnogi industrijski procesi kao što su toplinski tok i tok fluida su karakterizirani sporom aperiodskom dinamikom i mrtvim vremenom, i često se modeliraju procesnim modelom prvog reda plus mrtvo vrijeme (*first-order plus dead-time* – FOPDT), i u većini slučajeva su još uvijek upravljani proporcionalno-integracijsko-derivacijskim regulatorima. Među patentiranim i implemetiranim pristupima PID implementacije tzv. analitičke metode, tipično bazirane na procesima s otvorenom petljom ili zatvorenom petljom se obično preferiraju umjesto kompleksnijih metoda sinteze, kao što su metode bazirane na heurističkim pravilima, umjetnoj inteligenciji, i numeričkim optimizacijskim pristupima. U potonjim slučajevima ponašanje zatvorene petlje se tipično prati s uključenim PID regulatorom, a adaptacija regulatora se provodi u stvarnom vremenu bez primjene testnog signala. [11]

Konvencionale metode sinteze bazirane na formulama, kao što je Ziegler-Nicholsova (ZN) metoda, iako još uvijek u praktičnoj primjeni zbog svoje jednostavnosti, može rezultirati u relativno velikom nadvišenju kod skokovite pobude u zatvorenoj petlji, i relativno slabom prigušenju odziva.

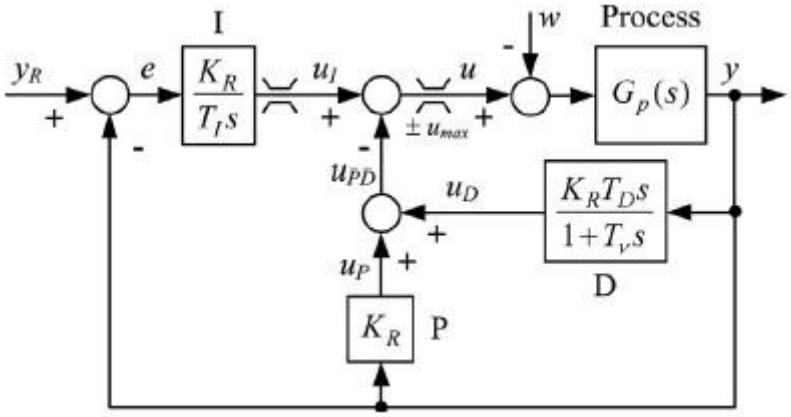
U većini slučajeva sinteza PID regulatora je bazirana na relativno jednostvnom FOPDT modelu procesa aproksimiran prvim redom Taylora ili Pade aproksimacijama mrtvog vremena koristeći u svrhu dizajniranja PID regulatora, koji možda neće biti točan kod kod dinamike sustava višeg reda. Da bi zadovoljili sustave višeg reda, dok istovremeno imamo relativno jednostavnu formulaciju procesa bez mrtvog vremena, može se koristiti PTn proces aperiodskog tipa, u kojem su analitički odnosi parametara između PTn modela i FOPDT modela tipično dani kroz odziv na skokovitu pobudu provlačenjem tangente kroz točku infleksije aperiodskog odziva ekvivalentnog procesnog modela.

Prema članku [11] ovdje je predložena metoda sinteze PID regulatora uz pomoć optimuma dvostrukog odnosa ODO (*damping optimum criterion*) u kombinaciji s PTn modelom procesa, koji olakšavaju analitički i izravan način podešavanja prigušenja i brzine odziva kod zatvorene petlje u odnosu na dinamiku procesa. Estimacija parametara PTn procesnog modela je ovdje bazirana na jednoj integraciji odziva procesa na skokovitu pobudu kako bi se pronašli parametri osnovnog FOPDT procesnog modela. Jednostavniji FOPDT

model je onda "srodan" ekvivalentnom PTn modelu bez mrtvog vremena, koristeći Taylorov razvoj višeg reda dinamike mrtvog vremena pomoću jednostavnih analitičkih izraza.

6.1. Struktura sustava upravljanja

Struktura linearnog kontrolnog sustava koji se sastoji od modificiranog PID regulatora (struktura I + PD) je prikazana na Slika 54. Kod tradicionalnog PID kontrolera, proporcionalni (P), integralni (I), dervacijski (D), članovi su smješteni na putanju regulacijske pogreške *e*, dok kod modificirane izvedbe proporcionalni i derivacijski član djeluju samo na izlaznu veličnu procesa y. Koristeći modificiranu izvedbu PID regulatora koji ne pridodaje dodatne nule u prijenosnu funkciju zatvorene petlje, moguće je postići željeno ponašanje sustava zatvorene petlje s obzirom na vanjski poremećaj w, dok istrovremeno izbjegavamo pretjerani napor regulatora zbog naglih promjena reference y_R ili šuma u referenci. Predložena struktura sustava upravljanja nema dodatnih nula zbog P + D djelovanja regulatora, predloženi pristup rezultira u relativno jednostavnim i izravnim pravilima sinteze PID regulatora. [11]



Slika 54. Blokovski dijagram sustava upravljanja s modificiranim PID regulatorom [11]

Ako bi se kojim slučajem koristila tradicionalna struktura PID regulatora, nešto kompleksnije metode sinteze bi bile potrebne.

Da bi bi se izbjeglo relativno veliko nadvišenje odziva na skokovitu pobudu kod velikih signala, na izlazu I člana je potrebno staviti limiter, a onda se limitira izlaz regulatora koristeći takozvanu metodu resetiranja integratora.

Pretpostavlja se da je proces karakteriziran relativno sporom aperiodskom dinamikom (bez nadvišenja na skokovitu pobudu i prolaznih oscilacija), koju je moguće aproksimirati aperiodnim modelom procesa n-tog reda s jednom vremenskom konstantom T_p (takozvani PTn model), danom u sljedećoj kompaktnoj formi prijenosne funkcije:

$$G_p(s) = \frac{y(s)}{u(s)} = \frac{K_p}{(1 + T_p s)^n} = \frac{K_p}{1 + a_1 s + a_2 s^2 + \cdots + a_n s^n} \tag{8}$$

gdje je K_p pojačanje PTn modela, $n \geq 2$ je red sustava, i parametri modela sustava karakterističnog polinoma a_m ($m = 1 \ldots n$) su dani:

$$a_m = \binom{n}{m} T_p^m = \frac{n!}{m!\,(n-m)!} T_p^m \tag{9}$$

6.1.1. Sinteza PID regulatora

Odabrani postupak sinteze PID regulatora se bazira na optimumu dvostrukog odnosa. Ovo je analitička metoda slična *pole placementu*, prikladna za vremenski-kontinurani objekt upravljanja (proces) s regulatorom punog ili reduciranog reda, što rezultira izravnim analitičkim relacijama između parametara regulatora, parametara objekta upravljanja, i željenog prigušenja odziva preko karakterističnih parametra. Optimum dvostrukog odnosa se koristi kod sustava upravljanja gdje se prigušenje zatvorene petlje mora regulirati na precizan i izravan način (električni servo motori – pr. elektronička zaklopka ETC).

Proces sinteze kreće s određivanjem prijenosne funkcije zatvorene petlje na slici 51. uz pretpostavku idealiziranog PID regulatora (tipično, vremenska konstanta filtera derivacijskog člana $T_v \ll T_D$ tako da je se može zanemariti):

$$G_c(s) = \frac{y(s)}{y_R(s)} = \\ = \frac{1}{\left(1 + \frac{(1 + K_R K_p) T_I s}{K_R K_p} + \frac{(K_R K_p T_D + a_1) T_I s^2}{K_R K_p} + \frac{a_2 T_I s^3}{K_R K_p} + \cdots + \frac{a_n T_I s^{n+1}}{K_R K_p}\right)} \tag{10}$$

Karakteristični polinom sustava zatvorene petlje (10) može biti prerađena prema optimumu dvostrukog odnosa na sljedeći način:

$$G_c(s) = \frac{1}{A_c(s)} = \frac{1}{1 + T_e s + D_2 T_e^2 s^2 + D_3 D_2^2 T_e^3 s^3 \cdots + D_l D_{l-1}^2 \cdots D_2^{l-1} T_e^l s^l} \quad (11)$$

gdje je T_e ekvivalentna vremenska konstanta sustava zatvorene petlje, $D_2, D_3, \ldots, D_l$ su takozvani karakteristični odnosi, a l je red sustava zatvorene petlje ($l = n + 1$ u slučaju PID regulatora (10)).

Kad su svi karakteristični odnosi namješteni na takozvane "optimalne" vrijednosti $D_2 = D_3 = \cdots = D_l = 0.5$ (primjerice kod uporabe regulatora punog reda), sustav zatvorene petlje bilo kojeg reda l ima kvazi-aperiodski odziv na skokovitu pobudu karakteriziran nadvišenjem od približno 6% (nalik na sustav drugog reda s faktorom prigušenja $\zeta = 0.707$) i približnim vremenom porasta 1.8-2.1 T_e. Ova sinteza u zatvorenoj petlji se može smatrati optimalnom u slučajevima kad je kritično potrebno malo nadvišenje i relativno dobro prigušeno ponašanje, kao kod upravljanih električnih pogona i srodnog upravljanja servopogonima. Odabirom veće vrijednosti ekvivalentne vremenske konstante T_e, robustnost sustava je unaprijeđena i osjetljivost na šum je smanjena, ali za uzvrat se dobije sporiji odziv i odbacivanje vanjskog poremećaja je manje efektivno. Generalno, prigušenje odziva se podešava variranjem karakterističnih odnosa $D_2, D_3, \ldots, D_l$, kod kojih na prigušenje dominantne dinamike zatvorene petlje najviše utječe najdominantniji karakteristični odnos D_2. Ukoliko bi smanjli odnos D_2 na približno 0.35 dobio bi se najbrži aperiodski odziv na skokovitu pobudu bez nadvišenja. U drugu ruku, ako se D_2 poveća iznad 0.5 prigušenje sustava zavorene petlje se smanjuje. [11]

PID regulator ($r = 3$) proizvoljno može namjestiti samo karakteristične odnose D_2, D_3 i D_4. Prema tome, analitički izrazi PID regulatora za proporcionalno pojačanje K_R, integralnu vremensku konstantu T_I i derivacijsku vremensku konstantu T_D se dobiju izjednačavanjem koeficijentima nižeg reda karakterističnog polinoma (10) s koeficijnetima karakterističnog polinoma (11) do s^4, koji nakon manipulacije i preuređivanja daju sljedeće relativno jednostavne analitičke izraze:

$$K_R = \frac{1}{K_p}\left(\frac{n(n-1)T_p^2}{2D_2^2 D_3 T_e^2} - 1\right) \tag{12}$$

$$T_I = \left(1 - \frac{2D_2^2 D_3 T_e^2}{n(n-1)T_p^2}\right)T_e \tag{13}$$

$$T_D = D_2 T_e T_p n \frac{(n-1)T_p - 2D_2 D_3 T_e}{n(n-1)T_p^2 - 2D_2^2 D_3 T_e^2} \tag{14}$$

s ekvivalentnom vremenskom konstantom T_e danom kako slijedi (vrijedi za $n > 2$):

$$T_e = \frac{(n-2)T_p}{3D_2 D_3 D_4} \tag{15}$$

Slično, za slučaj PI regulatora ($T_D = 0$), moguće je podesiti samo karakteristične odnose D_2, D_3, prema tome dizajn je baziran na koeficijentima karakterističnog polinoma (11) do s^3:

$$K_R = \frac{1}{K_p}\left(\frac{nT_p}{D_2 T_e} - 1\right) \tag{16}$$

$$T_I = \left(1 - \frac{D_2 T_e}{nT_p}\right)T_e \tag{17}$$

s ekvivalentnom vremenskom konstantom T_e danom kako slijedi (vrijedi za $n > 1$):

$$T_e = \frac{(n-1)T_p}{2D_2 D_3} \tag{18}$$

Gornji izrazi ukazuju na to da na parametre PID regulatora K_R, T_I i T_D direktno utječu dominantni karakteristični odnosi D_2 i D_3, dok nedominantni karakteristični odnos D_4 utječe samo na ekvivalentnu vremensku konstantu T_e zatvorene petlje. Slično, na sintezu jednostavnijeg PI regulatora direktno utječe najdominatniji karakteristični odnos D_2, dok na ekvivalentnu vremensku konstantu T_e zatvorene petlje direktno utječe manje dominantni karakteristični odnos D_3. Prema tome, kod zatvorene petlje brzina odziva i prigušenje se odvojeno podešavaju, zato što na podešenje prigušenja dominantne dinamike regulacijskog

kruga najviše utječe odabir karakterističnog odnosa D_2, dok na brzinu odziva direktno utječe ekvivalentna vremenska konstanta T_e. [11]

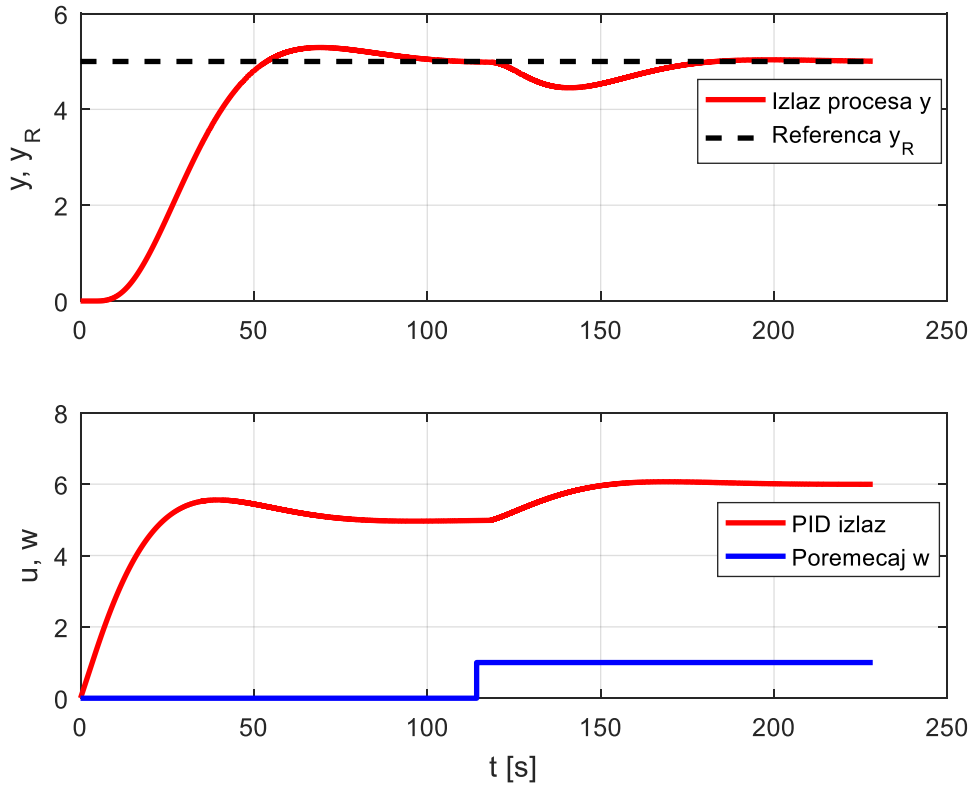
Za potrebe ovog zadatka, parametre regulatora (12) – (18) ću izračunati za odabrani red sustava $n = 4$, mrtvo vrijeme $T_t = 4\ s$, vremenska konstanta $T_p = 5{,}37\ s$, i jedinično pojačanje regulatora $K_p = 1$, (podatci uzeti iz literatura [11]). Karakteristič odnosi će biti postavljeni na optimalan iznos $D_2 = D_3 = D_4 = 0.5$. Rezultati parametara s odabranim vrijednostima niže u tablici:

Tablica 10. Izračunati parametri PID i PI regulatora za odabrani slučaj

	PID	PI
T_e	28.64 s	32.22 s
K_R	0.6875	0.3333
T_I	11.6681 s	8.055 s
T_D	3.9055 s	0

6.1.2. Simulacijska provjera sinteze regulatora

Svojstva predloženog načina sinteze PID regulatora su ilustrirana na Slika 55. simulacijskom provjerom. Slika 55. prikazuje simulacijske odzive vremenski-kontinuiranog objekta upravljanja s PID regulatorom podešenim za konzervativno prigušenje, kvazi-aperiodskog odziva ($D_2 = D_3 = D_4 = 0.5$) i PTn modela procesa karakteriziranog s $K_p = 1$, $T_p = 5{,}37\ s$ i reda sustava $n = 4$. Rezultati ukazuju na to da predložena metoda sinteze regulatora sa svim karaterističnim odnosima postavljenim na 0.5 stvarno daje dobro prigušen odziv upravljanog procesa u odnosu na skokovite promjene reference y_R i ulaznog poremećaja w, karakteriziran približnim nadvišenjem od 6%.



Slika 55. Odziv na skokovitu promjenu reference i poremećaja upravljanog procesa s PID regulatorom za procesni model PT4

6.2. Identifikacija modela procesa

Ovo potpoglavlje prezentira estimaciju parametra aperiodskog modela procesa baziranim na integraciji odziva na skokovitu pobudu ekvivalentnog FOPDT modela procesa (takozvana *area* metoda). Predložena identifikacija modela procesa je uspoređena s tradicionalnom metodom provlačenja tangente kroz točku infleksije aperiodskog odziva procesa estimacije parametara FOPDT i PTn procesnih modela, i koristi se kao osnova za *auto-tuning* algoritam PID regulatora. [11]

6.2.1. Postupak provlačenja tangente kroz točku infleksije aperiodskog odziva procesa (flexion-tangent pristup)

Tradicionalno, procesi karakterizirani aperiodskim odzivom na skokovitu pobudu i značajnim inicijalnim kašnjenjem (ekvivalent mrtvo vrijeme) su modelirani koristeći ekvivalentni model procesa prvog-reda plus mrtvo-vrijeme (FOPDT):

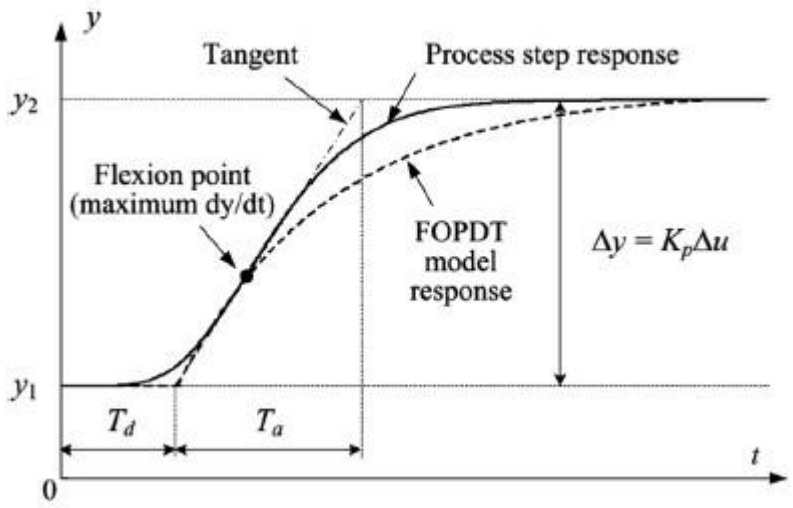
$$G_p(s) = \frac{K_p e^{-sT_d}}{1 + T_a s} \tag{19}$$

gdje su mrtvo vrijeme T_d i vremenska konstanta T_a tipično određeni metodom provlačenja tangente kroz točku infleksije pristupom kako je prikazano na Slika 56. Da bi dobili prikladniji procesni model bez mrtvog vremena (prikladan za sintezu PI/PID regulatora opisanu u potpoglavlju 6.1.1.), procesni model FOPDT može biti u direktnoj korelaciji s preciznijim procesnim modelom PTn (8), karakteriziran jednakim nagibom odziva (derivacija po vremenu) na točki infleksije (tzv. Strejc metoda). Odnosi između parametara FOPDT modela i PTn modela procesa su dani pomoću:

$$\frac{T_d}{T_a} = e^{(1-n)}\left[\frac{(n-1)^n}{(n-1)!} + \sum_{m=0}^{n-1}\frac{(n-1)^m}{m!}\right] - 1 \tag{20}$$

$$\frac{T_p}{T_a} = \frac{(n-1)^{n-1}}{(n-1)!}e^{(1-n)} \tag{21}$$

Preciznost ovako dobivenog FOPDT modela je povoljna samo u ekvivalentnom intervalu mrtvog vremena T_d i u blizini točke infleksije odziva sustava, dok je ostatak odziva procesa manje precizan, što se može vidjeti na Slika 56. [11]



Slika 56. Ilustracija identifikacije FOPDT modela baziranog na skokovitoj pobudi *flexion-tangent* pristupa [11]

6.2.2. Identifikacija FOPDT modela procesa baziranog na integraciji odziva na skokovitu pobudu

Zato što bi prethodna *flexion-tangent* metoda mogla biti osjetljiva na šum rezultata mjerenja procesa (kad se traži maksimalna točka odziva pomoću derivacije po vremenu), FOPDT model se može identificirati preko integracije po vremenu odziva na skokovitu pobudu, prikazano na Slika 57. U ovom pristupu, vremenska integracija odziva procesa na skokovitu pobudu je povezana s odzivom FOPDT modela na sljedeći način:

$$I_1 = \int_0^{T_{fin}} (y(t) - y_1)dt \approx \int_0^{T_{fin}} (y_2 - y_1)\left[1 - e^{-\frac{t-T_d}{T_a}}\right]dt =$$

$$= (y_2 - y_1)\left[T_{fin} - T_d - T_a + T_a e^{-\frac{(T_{fin}-T_d)}{T_a}}\right] \quad (22)$$

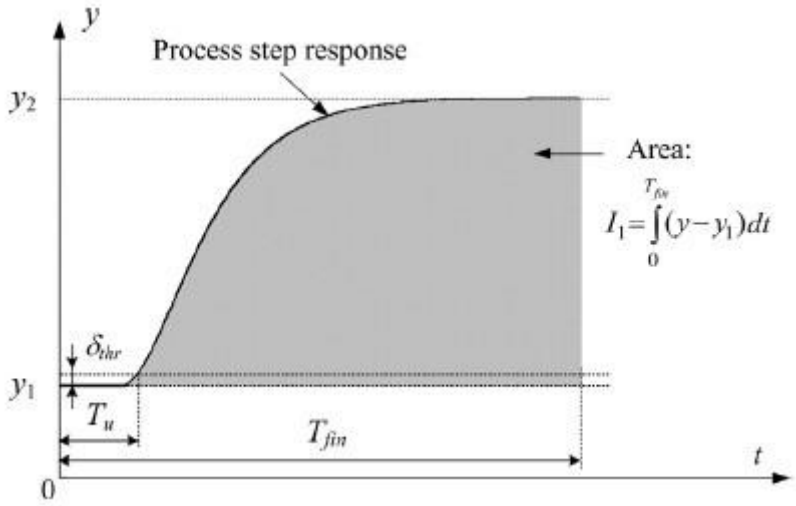
Ako je identifikacijski interval T_{fin} veći od $5T_a + T_d$ (stacionarno stanje), eksponencijalni pojam na desnoj strani jednadžbe (22) postaje zanemarljiv, pa vrijedi sljedeći odnos:

$$T_a = T_{fin} - T_d - \frac{I_1}{y_2 - y_1} \quad (23)$$

Ekvivalentno mrtvo vrijeme T_d modela procesa se može odrediti jednostavnom logikom prelaskom praga, kako je prikazano na Slika 57., dok je estimacija pojačanja K_p bazirana na stacionarnim stanjima procesa y_2 i y_1 i magnitude skokovite pobude Δu, kao u (24):

$$K_p = \frac{(y_2 - y_1)}{\Delta u} \quad (24)$$

U praktičnim aplikacijama, gornja identifikacija FOPDT procesnog modela bi trebala uključivati još neke značajke, a to je detaljno opisano u članku [11].



Slika 57. Ilustracija integracijske metode procesnog odziva za identifikaciju FOPDT modela [11]

6.2.3. Evaluacija ekvivalentnih parametara PTn modela

No ipak, mrtvo vrijeme T_d i vremenska konstanta prigušenja T_a FOPDT modela dobiveni gornjom identifikacijskom procedurom se prema (20) i (21) ne mogu koristiti za direktno računanje parametara PTn modela, zato što ekvivalent nagiba kod *flexion-tangent* više ne stoji. Da bi se pronašli parametri za model procesa višeg reda bez mrtvog vremena, kašnjenje je moguće aproksimirati Taylorovom ekspanzijom [11]:

$$e^{-sT_d} = \frac{1}{e^{sT_d}} \approx \frac{1}{1 + T_d s + \left(\frac{T_d^2 s^2}{2}\right) + \left(\frac{T_d^3 s^3}{6}\right) + \cdots} \tag{25}$$

što rezultira u sljedećoj aproksimaciji FOPDT modela:

$$G_p(s) = \frac{K_p}{1 + (T_d + T_a)s + \left(\left(\frac{T_d}{2}\right) + T_a\right) T_d s^2 + \left(\left(\frac{T_d}{6}\right) + \left(\frac{T_a}{2}\right)\right) T_d^2 s^3 + \cdots} \tag{26}$$

Gornji model bez mrtvog vremena može biti u relaciji s kompaktnim PTn modelom (8) prikladno korištenim u projektiranju PID regulatora, izjednačavanjem prva tri (dominantna) koeficijenta karakterističnog polinom PTn modela (8) s onima od

aproksimiranog modela procesa (26). Nakon nešto manipulacija i preuređivanja, dobije se niži izraz PTn modela reda n (zaokruži se na najbližu cjelobrojnu vrijednost):

$$n = \frac{2}{1 - \frac{T_d(T_d + 3T_a)}{(T_d + T_a)(T_d + 2T_a)}} \tag{27}$$

Slično, odnos između vremenske konstante T_a PTn modela i mrtvog vremena T_d i vremenske konstante prigušenja T_a FOPDT modela, dobije se sljedeći izraz (vrijedi za n > 2):

$$T_p = \sqrt{\frac{T_d(T_d + T_a)(T_d + 3T_a)}{n(n-2)(T_d + 2T_a)}} \tag{28}$$

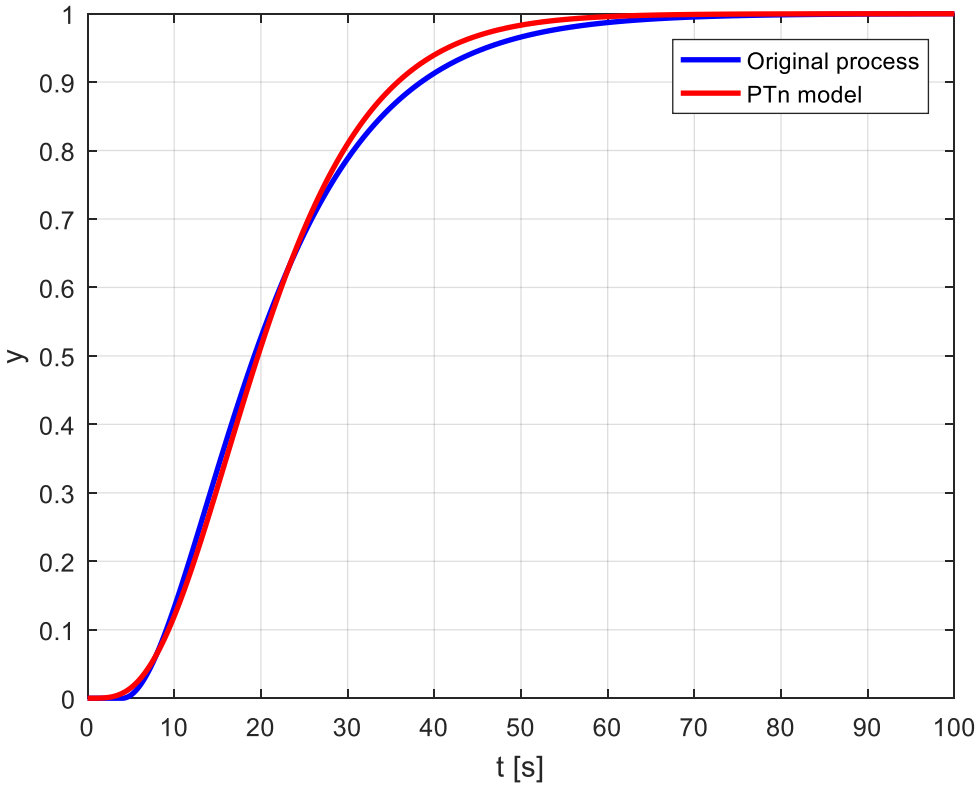
6.3. Rezultati simulacije predloženog PT4 modela procesa

Rezultati identifikacije predloženog PTn modela procesa su prikazani *area* metodom za slučaj modela procesa koji se sastoji aperidoske dinamike, mrtvog vremena i relativno nenaglašene nule u prijenosnoj funkciji:

$$G_p(s) = \frac{y(s)}{u(s)} = \frac{(1 + T_4 s)e^{-T_t s}}{(1 + T_1 s)(1 + T_2 s)(1 + T_3 s)} \tag{29}$$

kod koje su vrijednosti vremenskih konstanti $T_1 = 3\ s,\ T_2 = 7\ s,\ T_3 = 10\ s,\ T_4 = 2\ s$ i vremenska konstanta mrtvog vremena $T_t = 4\ s$. [11]

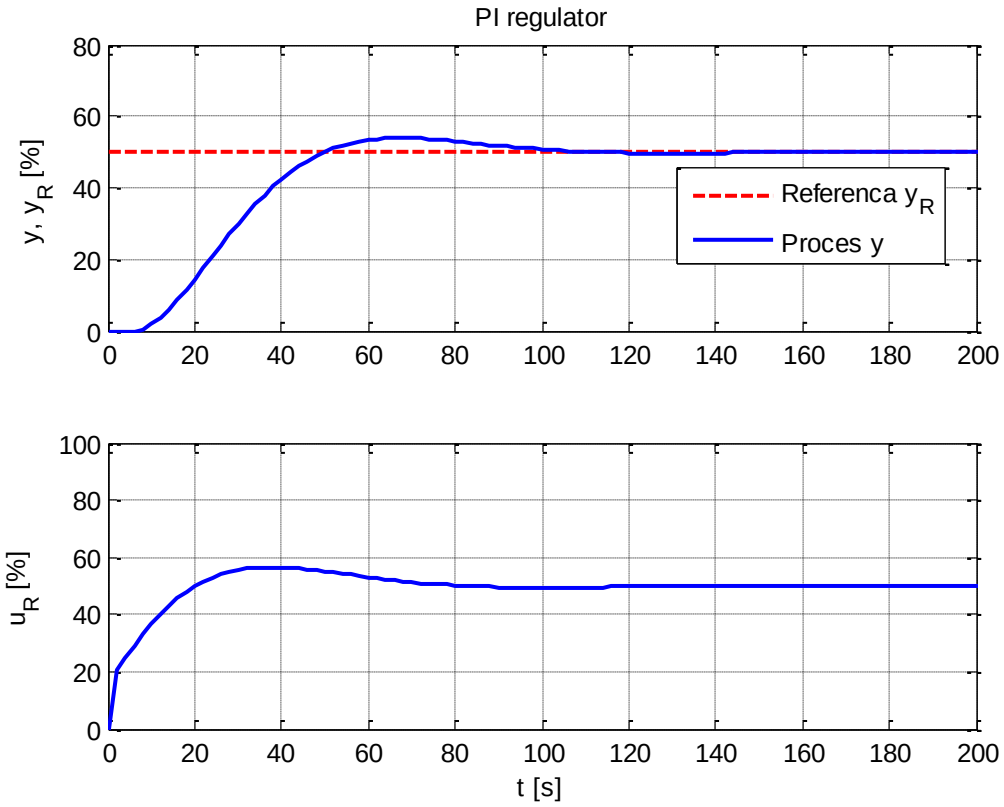
Slika 58. prikazuje simulaciju identifikacije PTn modela *area* metodom za gornji slučaj.



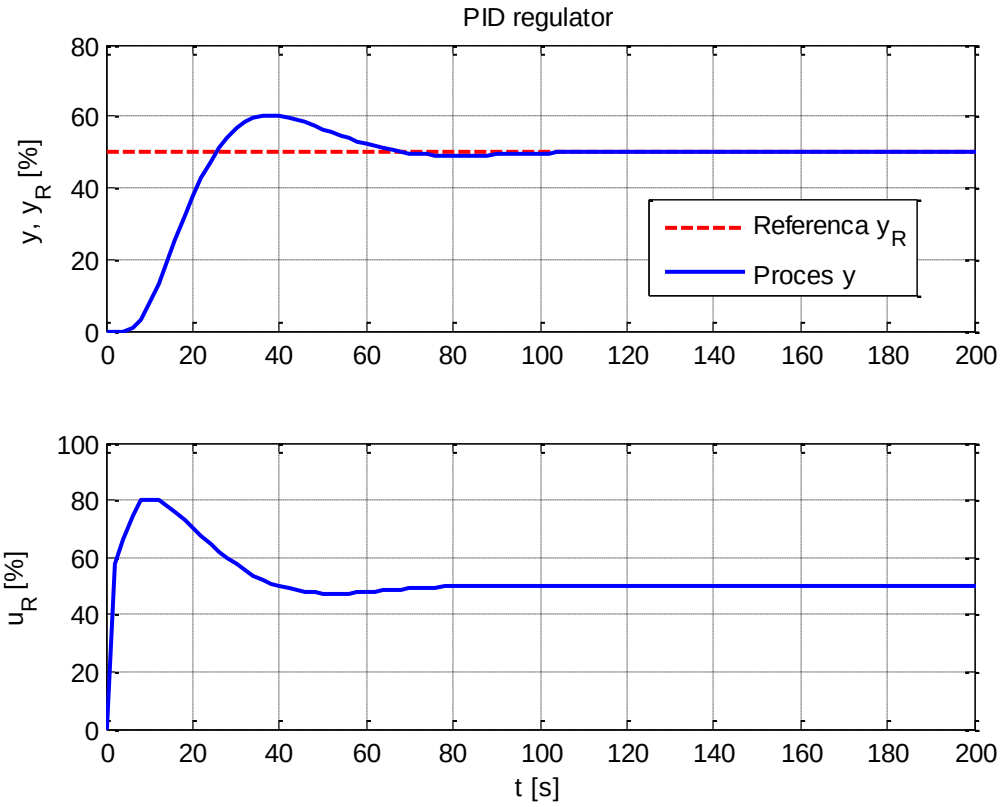
Slika 58. Simulacija identifikacije PTn modela dobivenog *area* metodom za mrtvo vrijeme iznosa Tt = 4 s

6.4. Rezultati simulacije PID regulatora iz PLC-a

Simulacija rada PID regulatora implementiranog u S7-200 se provela na modelu koji se koristio u auditornim vježbama iz kolegija Mikroprocesorsko upravljanje (literatura [10]), čiji je ljestvičasti dijagram u prilogu ovog rada (listing programa). Izračunate vrijednosti parametara koji su zapisani u tablici 9. se unose u podatkovni blok (Data Block) za inicijalizaciju globalnih varijabli (V memory). PLC se postavi u radni (RUN) mod i dobiju se rezultati testa. Nakon što PLC odradi simulaciju, dobivene podatke iz tablice stanja (Status Chart) koja služi za on-line nadzor programa i podataka u memoriji, kopiraju se u tekstualnu datoteku (.txt) odakle se poziva preko M funkcije Matlaba i dobiju se rezultati simulacije prikazani na slikama.



Slika 59. Rezultati simulacije PI regulatora



Slika 60. Rezultati simulacije PID regulatora

Radi usporedbe djelovanja provedena je simulacija PI i PID regulatora. Očito je da PID regulator ima značajno brži odziv, potrebno je dvostruko manje vremena do prvog prolaska kroz stacionarno stanje, t_{rPI+D} = 25,3 [sec] naspram t_{rPI} = 50 [sec] koliko je potrebno PI regulatoru, i vrijeme trajanja smirivanja prijelazne pojave je kraće kod PID regulatora nego kod PI regulatora. Isto tako se može primjetiti da je nadvišenje kod PID regulatora dosta veće nego kod PI regulatora, σ_{mPI+D} = 20% naspram σ_{mPI} = 8%.

Uspoređujući projektirani I+PD regulator u (potpoglavlje 6.1.) s regulatorom PI+D iz CPU jedinice PLC-a može se primjetiti da je brzina odziva I+PD regulatora manja od PI+D regulatora. Također, nadvišenje I+PD je dosta manje σ_{mI+PD} = 6% ali odabirom svih karakterističnih odnosa na optimalan iznos = 0.5 postiže se tzv. kvazi-aperiodski odziv regulacijskog kruga sa 6% nadvišenja u odzivu.

7. Zaključak

Regulacija temperature i protok fluida su često karakterizirani sporom aperiodskom dinamikom i mrtvim vremenom, i najčešće su modelirani uz pomoć sustava prvog reda s mrtvim vremenom (FOPDT). U ovom radu je predstavljena analitička metoda sinteze PID regulatora temeljena na vremenski-kontinuiranom modelu objekta upravljanja dobivenog postupkom identifikacije dinamičkog modela procesa aperiodskog tipa (PTn model), i upotreba optimuma dvostrukog odnosa za određivanje parametara regulatora čije glavne značajke su utvrđene uz pomoć simulacija.

Da bi se pojednostavnila sinteza PID regulatora kod modela procesa aperiodskog tipa odabrana je I + PD struktura regulatora gdje proporcionalni i derivacijski član djeluju na izlaznu veličinu y a ne na veličinu pogreške sustava e. Na taj način su se izbjegle dodatne nule u prijenosnoj funkciji regulatora. Izvedeni su izrazi za određivanje parametara regulatora uz pomoć optimuma dvostrukog odnosa i vidljivo je da iznosi karakterističnih odnosa imaju velik utjecaj na iznose parametara. Za potrebe ovog primjera karakteristični odnosi su postavljeni na 0.5, na kojima sustav bilo kojeg reda ima kvazi-aperiodski odziv na skokovitu pobudu karakteriziran nadvišenjem od približno 6%, što je dokazano provedenom simulacijom.

Prema članku [11] izvršena je identifikacija predloženog modela procesa PTn za mrtvo vrijeme $T_t = 4\ s$ i dobiveni su zadovoljavajući rezultati, odnosno PTn model dobro slijedi izvorni odziv modela procesa. Izračunati parametri regulatora su implementirani u PLC i provedena je analiza PI i PID regulatora. PID regulator ima značajno brži odziv od PI regulatora ali i veće nadvišenje. Isto tako uspoređujući PID regulator iz PLC sa projektiranim regulatorom prema modelu procesa lako se primjeti da ovaj regulator ima brži odziv i brže vrijeme smirivanja ali je i nadvišenje dosta veće. Razlog je taj što je PID regulator u PLC-u strukture PI+D, odnosno ima izraženiju nulu u prijenosnoj funkciji što smo nastojali izbjeći kod projektiranog regulatora da bi se pojednostavnila sinteza. No regulator u PLC-u ne pokazuje rezidulane oscilatornosti u stacionarnom dijelu odziva tako da su polovi sustava podešeni za dobru prigušenost odziva što i je svojstvo optimuma dvostrukog odnosa.

Sljedeći korak bi bio spajanje fizičkih ulaza na registre stanja logičkih ulaza i spajanje fizičkih izlaza na registre stanja logičkih izlaza kako bi se provela eksperimentalna provjera

na realnom objektu opisanom (fizičkom modelu procesa) karakteriziranom aperiodskom dinamikom i mrtvim vremenom.

LITERATURA

[1] Lamb, F.: Industrial Automation Hands-On, McGraw-Hill Education New York, 2013.

[2] Stenerson, J.: Industrial Automation and Process Control, Prentice Hall, Upper Saddle River, New Jersey, 2003.

[3] Zhang, P.: Industrial Control Technology – A Handbook for Engineers and Researchers, William Andrew Norwich, 2008.

[4] Bolton, W.: Programmable Logic Controllers – Fourth Edition, Elsevier Oxford, 2006.

[5] Kamel, K.; Kamel, E.: Programmable Logic Controllers – Industrial Control, McGraw-Hill Education New York, 2014.

[6] https://library.automationdirect.com/history-of-the-plc/ (17.02.2019.)

[7] https://new.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-1200.html (23.02.2019.)

[8] Siemens AG: Simatic S7-200 Programmable Controller System Manual, Nuernberg, 2005.

[9] http://www.automation.siemens.com/ (03.03.2019.)

[10] Pavković, D.: *Auditorne vježbe iz kolegija "Mikroprocesorsko upravljanje"*, Fakultet strojarstva i brodogradnje Sveučilišta u Zagrebu, 2010.

[11] Pavković, D.; Polak, S.; Zorc, D.; *PID controller auto-tuning based on process step response and damping optimum criterion*, ISA Transactions, Vol. 53, No. 1, pp. 85-96, 2014.

PRILOZI

I. CD-R disc

II. Matlab kod za sintezu regulatora i identifikaciju procesnog modela

III. Matlab kod za simulaciju PI i PID regulatora

IV. Ljestvičasti dijagram (*Ladder Diagram*)

V. Tablica simbola (*Symbol Table*)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Identifikacija PTn modela primjenom metode integriranja odziva
%
%          y(s)          Kp
% Gp(s) = ------ = --------------
%          u(s)     (1 + Tp*s)^n
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Odabir tipa procesa za analizu


echo on

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                    %                                      %
% Proces 1:                          % Proces 2:                           %
%              (1 + 2s)exp(-4s)      %             (1 + 2s)exp(-4s)         %
% G(s) = -------------------------   % G(s) = -------------------------    %
%        (1 + 3s)(1 + 7s)(1 + 10s)   %        (1 + 3s)(1 + 7s)(1 + 10s)     %
%                                    %                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                    %                                      %
% Proces 3:                          % Proces 4:                           %
%             (1 + 2s)exp(-12s)      %             (1 + 2s)exp(-16s)        %
% G(s) = -------------------------   % G(s) = -------------------------    %
%        (1 + 3s)(1 + 7s)(1 + 10s)   %        (1 + 3s)(1 + 7s)(1 + 10s)     %
%                                    %                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

echo off

% Vremenski vektor
Ts = 0.01;
% Step djeluje u nultoj sekundi

while 1
        proc_type = input('Tip procesa = ');
        proc_type = fix(proc_type);

        if(proc_type == 1)
                t = 0:Ts:100.0;
                Kp_ = 1.0;
                num = Kp_*[0 0 2 1];
                den = conv([3 1],conv([7 1],[10 1]));
                tfc = tf(num,den);
                yc_ = step(tfc,t);
                yc = yc_;
                % mrtvo vrijeme iznosa Td sek (Td = Nsmp*T)
                Td = 4.0;
                Nsmp = fix(Td/Ts);
                for(cnt = 1:Nsmp)
                        yc(cnt) = 0.0;
                end
```

```
                for(cnt=Nsmp+1:length(t))
                        yc(cnt) = yc_(cnt-Nsmp);
                end

                break;
        elseif(proc_type == 2)
                t = 0:Ts:100.0;
                Kp_ = 1.0;
                num = Kp_*[0 0 2 1];
                den = conv([3 1],conv([7 1],[10 1]));
                tfc = tf(num,den);
                yc_ = step(tfc,t);
                yc = yc_;
                % mrtvo vrijeme iznosa Td sek (Td = Nsmp*T)
                Td = 4.0;
                Nsmp = fix(Td/Ts);
                for(cnt = 1:Nsmp)
                        yc(cnt) = 0.0;
                end

                for(cnt=Nsmp+1:length(t))
                        yc(cnt) = yc_(cnt-Nsmp);
                end

                break;
        elseif(proc_type == 3)
                t = 0:Ts:100.0;
                Kp_ = 1.0;
                num = Kp_*[0 0 2 1];
                den = conv([3 1],conv([7 1],[10 1]));
                tfc = tf(num,den);
                yc_ = step(tfc,t);
                yc = yc_;
                % mrtvo vrijeme iznosa Td sek (Td = Nsmp*T)
                Td = 12.0;
                Nsmp = fix(Td/Ts);
                for(cnt = 1:Nsmp)
                        yc(cnt) = 0.0;
                end

                for(cnt=Nsmp+1:length(t))
                        yc(cnt) = yc_(cnt-Nsmp);
                end

                break;
        elseif(proc_type == 4)
                t = 0:Ts:100.0;
                Kp_ = 1.0;
                num = Kp_*[0 0 2 1];
                den = conv([3 1],conv([7 1],[10 1]));
                tfc = tf(num,den);
                yc_ = step(tfc,t);
                yc = yc_;
```

```
                % mrtvo vrijeme iznosa Td sek (Td = Nsmp*T)
                Td = 16.0;
                Nsmp = fix(Td/Ts);
                for(cnt = 1:Nsmp)
                        yc(cnt) = 0.0;
                end

                for(cnt=Nsmp+1:length(t))
                        yc(cnt) = yc_(cnt-Nsmp);
                end

                break;
        else
                fprintf(1,'\n Pogresan tip procesa, ponovite unos \n');
        end
end

while 1
        noise_flag = input('Sum mjerenja? (0 - bez suma, 1 - sum postoji) = ');
        noise_flag = fix(noise_flag);
        stdev = 0.05*Kp_;

        if(noise_flag == 1)
                randn('state',0);
                % Standardna devijacija suma
                noise = stdev*randn(size(yc));
                ym = yc + noise;
                filter_flag = 1;
                break;
        elseif(noise_flag == 0)
                ym = yc;
                filter_flag = 0;
                break;
        else
                fprintf(1,'\n Pogreska unosa: ponovite unos! \n');
        end
end

% Vektor za spremanje filtriranih uzoraka
yfmean = zeros(size(t));

% Sirina "prozora" osrednjavajuceg filtra
W = 20; % Paran broj

% Racunanje i spremanje osrednjenih uzoraka
for(cnt = W+1:length(t))
        yfmean(cnt) = mean(ym(cnt-W:cnt));
end

% Trazenje iznosa mrtvog vremena
for(cnt = W+1:length(t))
        if(yfmean(cnt) > 1*stdev)
                indx = cnt - W/2;
                break;
```

```
        end
end

Tu = t(indx);

% Numericki proracun vrem. integrala odziva
% trapezna integracija
y_int = 0.0;
for(cnt = 2:length(t))
        y_int = y_int + 0.5*Ts*(ym(cnt) + ym(cnt-1));
end

% Stacionarno stanje procesa
yss = mean(ym(length(t)-20:length(t)));

% Potraga za mrtvim vremenom i nadomjesnom vremenskom konstantom
Tg_plus_Tu = t(length(t)) - y_int/yss;

% Za jedinicni skok pobude pojacanje je jednako promjeni stac. stanja procesa
Kp_est = yss/1.0;

Tg = Tg_plus_Tu - 1.0*Tu;

% Ispis parametara procesa
fprintf(1,'\n Nadomjesno mrtvo vrijeme Tu = %f \n', Tu);
fprintf(1,'\n Nadomjesna vr. konstanta Tg = %f \n', Tg);
fprintf(1,'\n Pojacanje procesa Kp = %f \n', Kp_est);

% Proracun parametara PTn modela procesa Tp i n

 n = ceil(2.0/(1.0 - 3.0*(Tg/2 + Tu/6)*Tu/(Tu/2 + Tg)/(Tu + Tg)));
 Tp = sqrt(3/n/(n-2)*(Tu/6 + Tg/2)*Tu*(Tu+Tg)/(Tg + Tu/2));

%n = round(2.0/(1.0 - (3*Tg + Tu)*Tu/(Tu + 2*Tg)/(Tu + Tg)));
%Tp = sqrt(1/n/(n-2)*(Tu + 3*Tg)*Tu*(Tu+Tg)/(2*Tg + Tu));

% Ispis parametara PTn procesa
fprintf(1,'\n Red procesa n = %d \n', n);
fprintf(1,'\n Nadomjesna vr. konstanta Tp = %f \n', Tp);
fprintf(1,'\n Pojacanje procesa Kp = %f \n', Kp_est);

num_n = Kp_est;
den_n = [Tp 1];
for(cnt = 1:n-1)
        den_n = conv(den_n,[Tp 1]);
end

% Comparison
figure(7),plot(t,ym,'b','LineWidth',2),hold on,grid on
figure(7),plot(t,yn_est,'r','LineWidth',2),hold on,grid on
xlabel('t [s]');ylabel('y');legend('Original process','PTn model',4)

num_nd = Kp_est;
den_nd = conv([(Tu^3)/6 (Tu^2)/2 Tu 1],[Tg 1]);
```

```
yn_est = step(tf(num_n,den_n),t);
ynd_est = step(tf(num_nd,den_nd),t);

y_dl_est = step(tf(num_n,[Tg 1]),t(1:length(t)-indx));

y_dl_calc = [zeros(size(1:indx)) y_dl_est']';

% Usporedba modela procesa
figure(1),plot(t,ym,'b','LineWidth',2),hold on,grid on
figure(1),plot(t,yn_est,'r','LineWidth',2),hold on,grid on
figure(1),plot(t,y_dl_calc,'g','LineWidth',2),hold on,grid on
xlabel('t [s]');ylabel('y');legend('Originalni proces','PTn model proc.','FOPDT
model',4)

MSEn = sqrt(sum((ym - yn_est).^2))/length(t);
fprintf(1,'\n Srednje kv. odstupanje odziva modela procesa MSEn = %f \n', MSEn);

MSEd = sqrt(sum((ym - y_dl_calc).^2))/length(t);
fprintf(1,'\n Srednje kv. odstupanje odziva FOPDT modela procesa MSEd = %f \n',
MSEd);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simulacijska provjera PID regulatora
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Vrijeme uzorkovanja << Tp, Td i Ta
Tp_ = Tp + Ts/n;
% Parametri PID regulatora
% Karakteristicni odnosi
D2 = 0.5; D3 = 0.5; D4 = 0.5;
% Ekvivalentna vremenska konstanta zatvorenog kruga
Te = (n - 2)*Tp_/3.0/D2/D3/D4;
% Pojacanje
KR = (n*(n - 1)*Tp_*Tp_/D2/D2/D3/Te/Te/2 - 1)/Kp_est;
% Integralna vremenska konstanta
TI = (1 - 2*D2*D2*D3*Te*Te/n/(n-1)/Tp_/Tp_)*Te;
% Derivacijska vremenska konstanta
TD = D2*Te*Tp_*n*((n-1)*Tp_ - 2*D2*D3*Te)/(n*(n-1)*Tp_*Tp_ - 2*D2*D2*D3*Te*Te);
% Limiti
umax = 10.0; umin = -1.0*umax;
% Trajanje simulacije
t_end = 8*Te;

PID_test_antiwindup;
sim('PID_test_antiwindup');

figure(2)
subplot(211),plot(t,y,'r','LineWidth',2),grid on,hold on
subplot(211),plot(t,yR,'k--','LineWidth',2),grid on,hold on
ylabel('y')
ylabel(' y, y_R '), legend('Izlaz procesa y', 'Referenca y_R',4)
subplot(212),plot(t,u,'r','LineWidth',2),grid on,hold on
```

```
subplot(212),plot(t,w,'b','LineWidth',2),grid on,hold on
ylabel(' u, w '), legend('PID izlaz', 'Poremecaj w',2)
xlabel('t [s]')
```

```
pv = load('PI_ctrl_PV.txt');
pv = pv/32000;
m = load('PI_ctrl_M.txt');
m = m/32000;
Ts = 2.0;
t = 0:Ts:149*Ts;
sp = 0.5*ones(size(t));

figure(1),
subplot(211),plot(t,100*sp,'r--','LineWidth',2),grid on,hold on
subplot(211),plot(t,100*pv,'b','LineWidth',2),grid on,hold on
ylabel('y, y_R [%]'),legend('Referenca y_R','Proces y')
title('PI regulator')
subplot(212),plot(t,100*m,'b','LineWidth',2),grid on,hold on
ylabel('u_R [%]'),xlabel('t [s]')

pv = load('PID_ctrl_PV.txt');
pv = pv/32000;
m = load('PID_ctrl_M.txt');
m = m/32000;
Ts = 2.0;
t = 0:Ts:149*Ts;
sp = 0.5*ones(size(t));

figure(2),
subplot(211),plot(t,100*sp,'r--','LineWidth',2),grid on,hold on
subplot(211),plot(t,100*pv,'b','LineWidth',2),grid on,hold on
ylabel('y, y_R [%]'),legend('Referenca y_R','Proces y')
title('PID regulator')
subplot(212),plot(t,100*m,'b','LineWidth',2),grid on,hold on
ylabel('u_R [%]'),xlabel('t [s]')
```

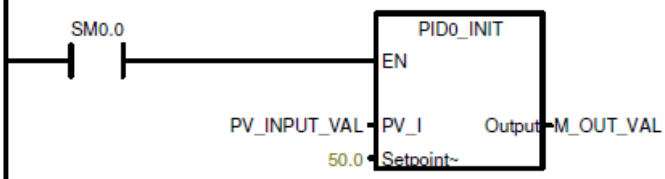
Labvj3_PIDEXE / MAIN (OB1)

Block: MAIN
Author:
Created: 11/06/2010 12:12:20 pm
Last Modified: 03/05/2019 01:32:09 pm

Symbol	Var Type	Data Type	Comment
	TEMP		
	TEMP		
	TEMP		
	TEMP		

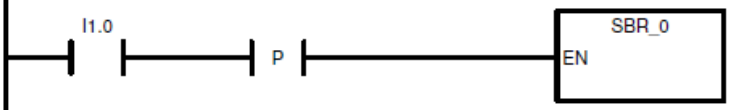
Testing integrated PID function

Network 1 Initialization of PID controller

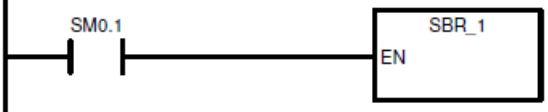


Symbol	Address	Comment
M_OUT_VAL	VW28	Controller output for Analog Out
PV_INPUT_VAL	VW16	Process variable from Analog IN

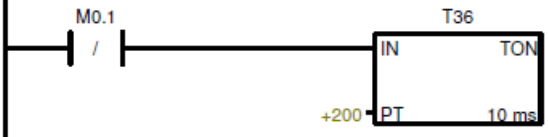
Network 2 Initialization of experiment: setting PID controller output and integral state to zero



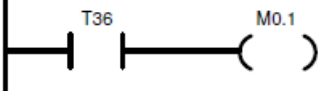
Network 3



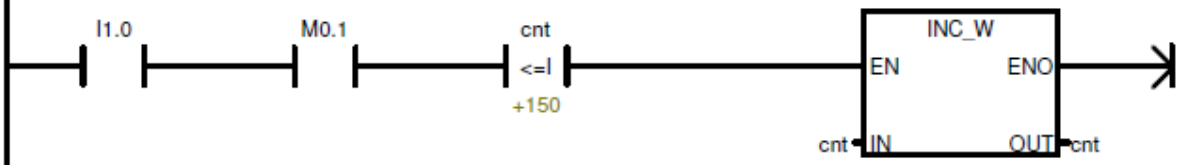
Network 4 Generation of additional time base for data logging (T = 2 s)



Network 5

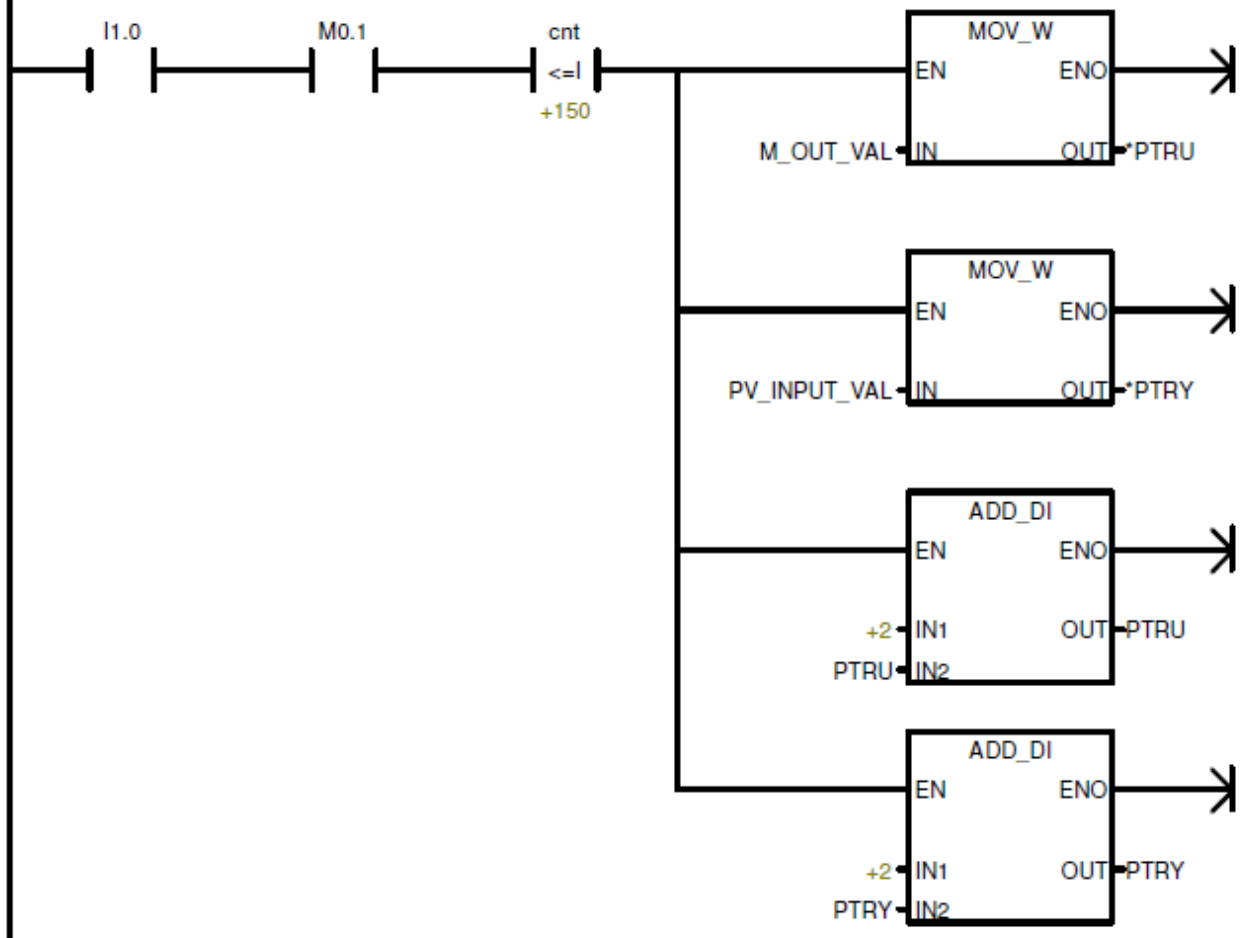


Network 6 Data logging counter (up to 150 samples, for sampling time T = 2.0 s => 300 s of data recording)



Symbol	Address	Comment
cnt	VW72	Counter for saving data

Network 7 Saving data in INT format (range 0 ... 32000) via pointers



Symbol	Address	Comment
cnt	VW72	Counter for saving data
M_OUT_VAL	VW28	Controller output for Analog Out
PTRU	VD64	Pointer No. 1 (controller output)
PTRY	VD68	Pointer No. 2 (process output)
PV_INPUT_VAL	VW16	Process variable from Analog IN

Labvj3_PIDEXE / SBR_0 (SBR0)

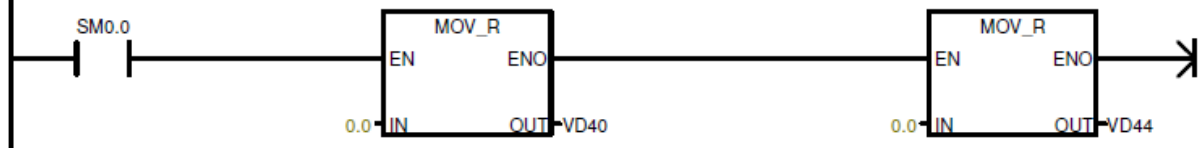
Block: SBR_0
Author:
Created: 11/16/2010 09:19:55 am
Last Modified: 02/20/2019 03:00:18 pm

Symbol	Var Type	Data Type	Comment
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

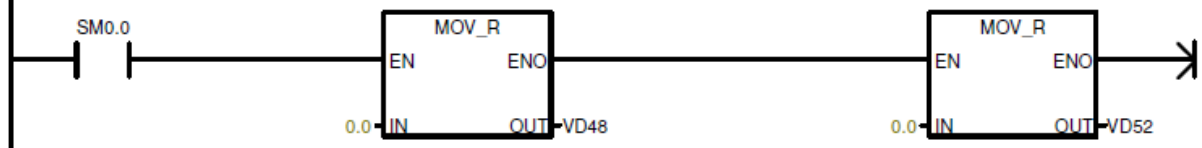
POU Comment

Network 1 Network Title

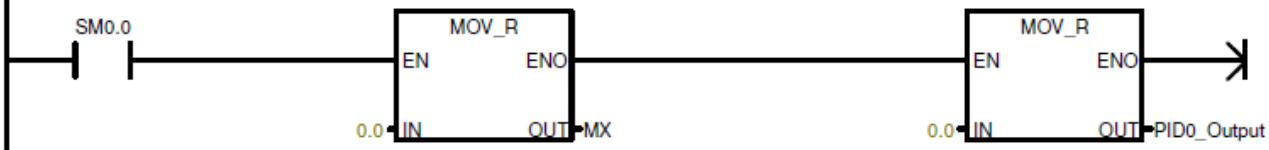
Network Comment



Network 2

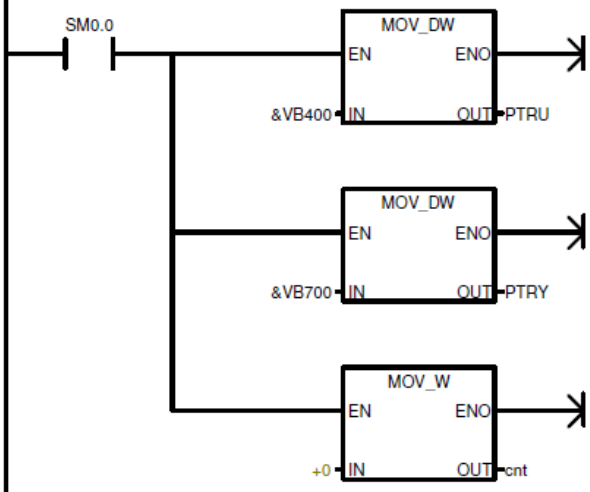


Network 3



Symbol	Address	Comment
MX	VD128	
PID0_Output	VD108	Calculated, Normalized Loop Output

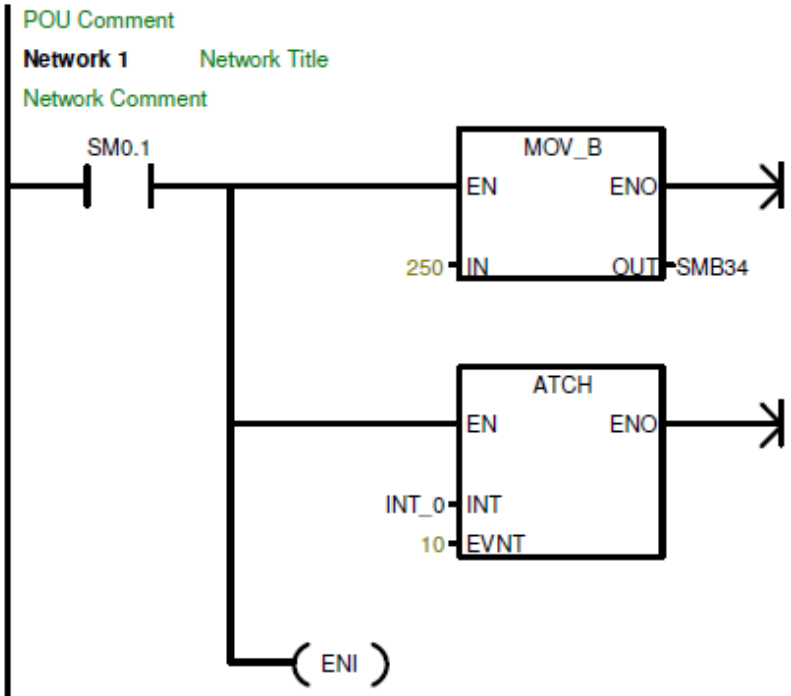
Network 4



Symbol	Address	Comment
cnt	VW72	Counter for saving data
PTRU	VD64	Pointer No. 1 (controller output)
PTRY	VD68	Pointer No. 2 (process output)

Block: SBR_1
Author:
Created: 11/15/2010 05:34:04 pm
Last Modified: 02/20/2019 03:51:51 pm

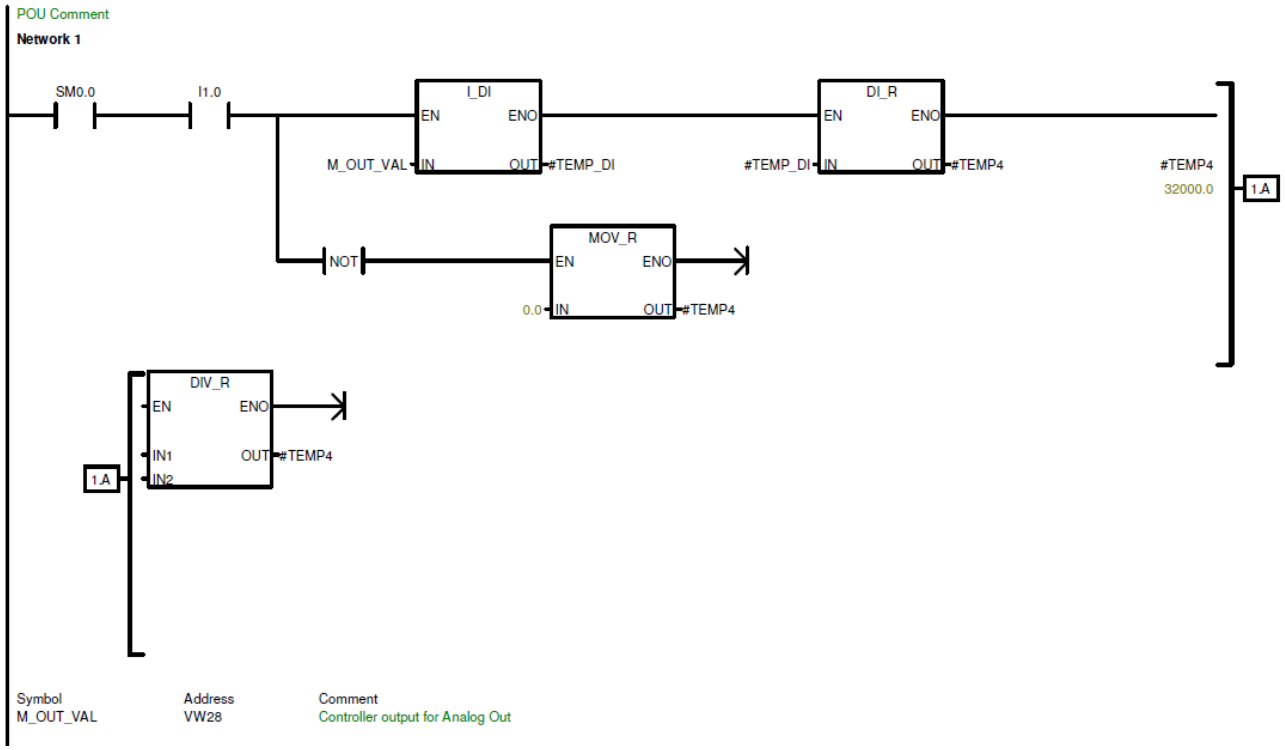
Symbol	Var Type	Data Type	Comment
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

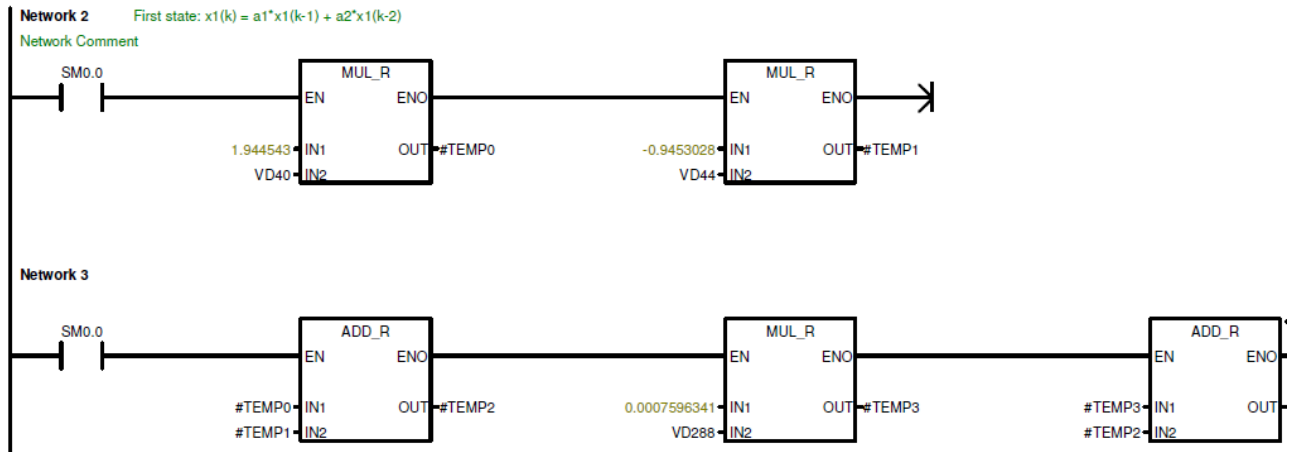


Symbol	Address	Comment
INT_0	INT0	POU Comment

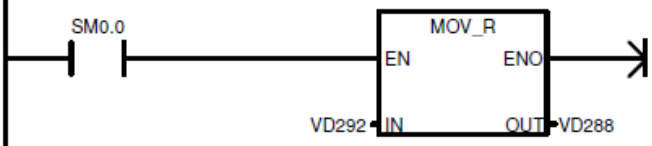
Block: INT_0
Author:
Created: 11/15/2010 05:28:40 pm
Last Modified: 03/05/2019 01:32:09 pm

	Symbol	Var Type	Data Type	Comment
LD0	TEMP0	TEMP	REAL	
LD4	TEMP1	TEMP	REAL	
LD8	TEMP2	TEMP	REAL	
LD12	TEMP3	TEMP	REAL	
LD16	TEMP4	TEMP	REAL	
LD20	TEMP_DI	TEMP	DINT	
LD24	TEMP5	TEMP	REAL	

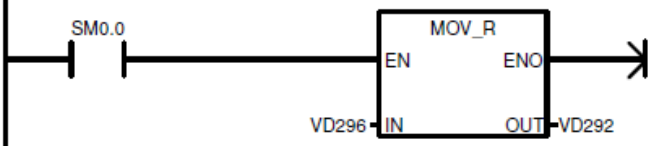




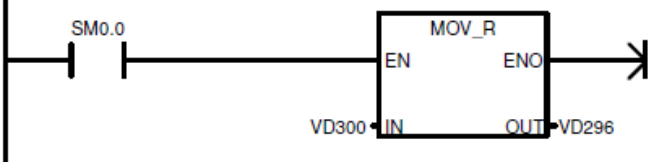
Network 4 Update dead-time states (1)



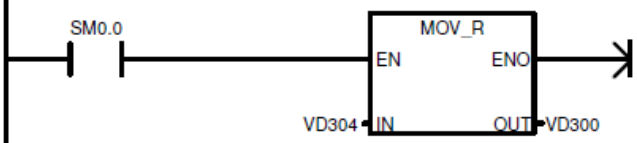
Network 5 Update dead-time states (2)



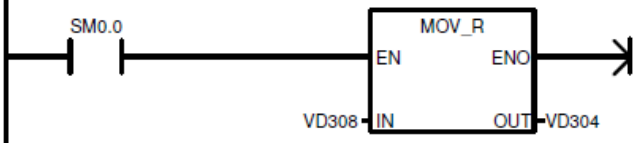
Network 6 Update dead-time states (3)



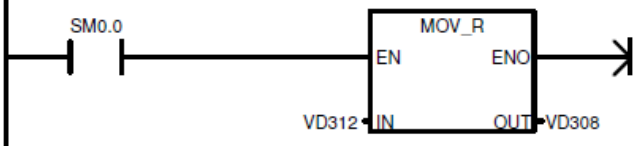
Network 7 Update dead-time states (4)



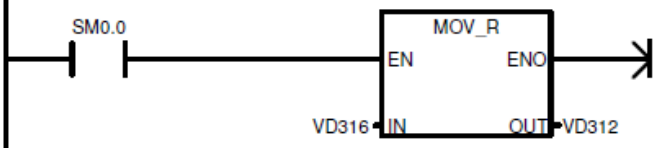
Network 8 Update dead-time states (5)



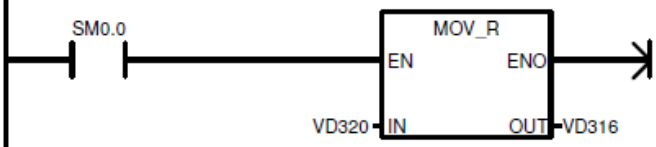
Network 9 Update dead-time states (6)



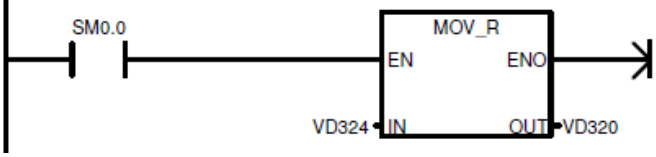
Network 10 Update dead-time states (7)



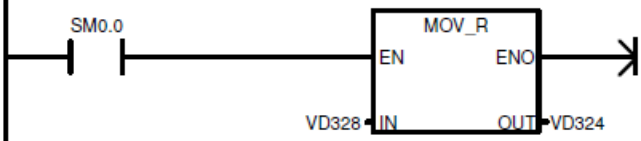
Network 11 Update dead-time states (8)



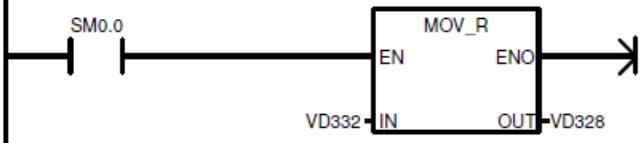
Network 12 Update dead-time states (9)



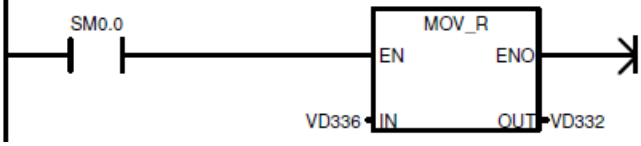
Network 13 Update dead-time states (10)



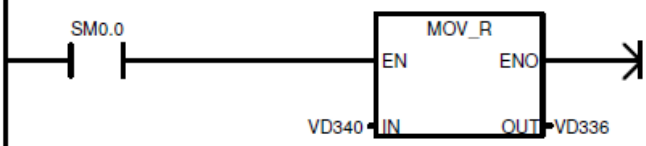
Network 14 Update dead-time states (11)



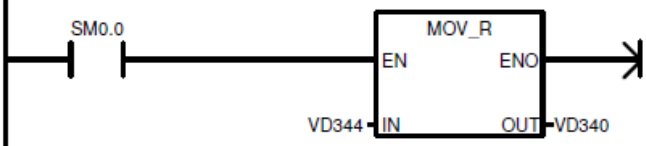
Network 15 Update dead-time states (12)



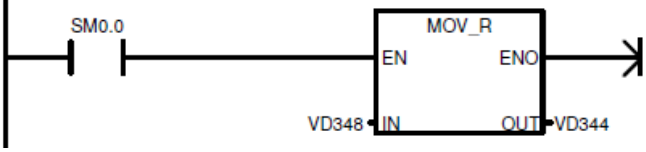
Network 16 Update dead-time states (13)



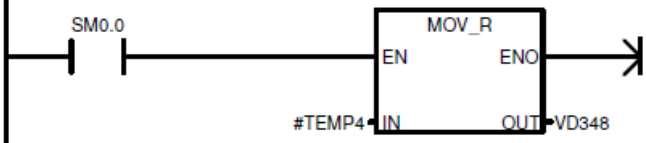
Network 17 Update dead-time states (14)



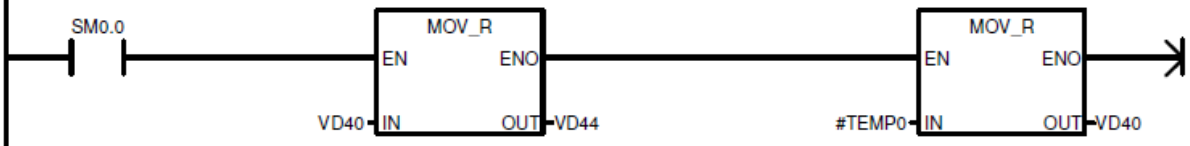
Network 18 Update dead-time states (15)



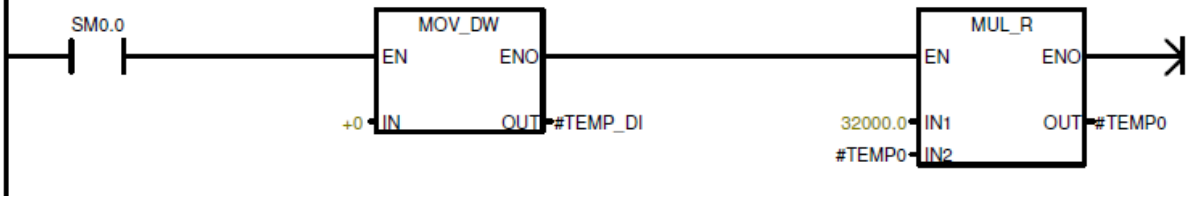
Network 19 Update dead-time states (16)

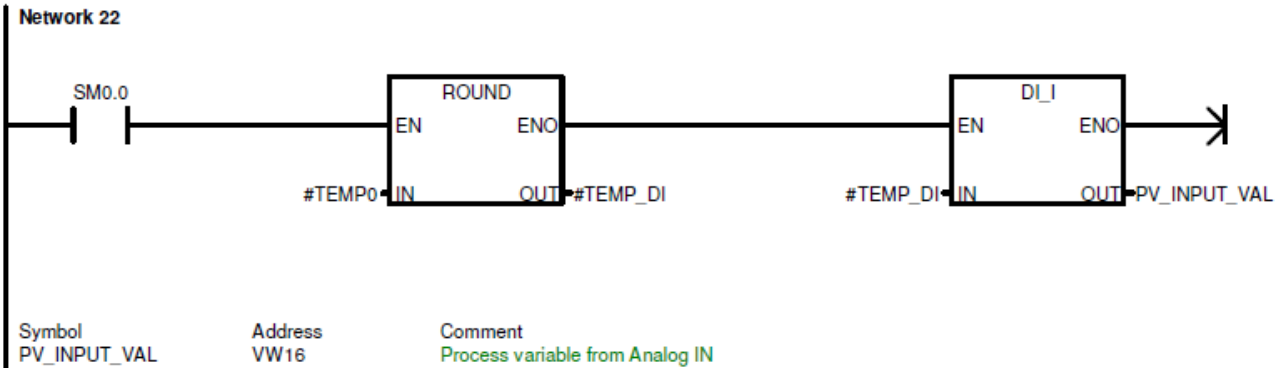


Network 20 Update process states and store process value for PID controller



Network 21




Network 22
SM0.0
ROUND
EN
ENO
#TEMP0
IN
OUT
#TEMP_DI
DI_I
EN
ENO
#TEMP_DI
IN
OUT
PV_INPUT_VAL
Symbol
Address
Comment
PV_INPUT_VAL
VW16
Process variable from Analog IN

Labvj3_PIDEXE / PID0_INIT (SBR2)

Block: PID0_INIT
Author: S7-200 Instruction Wizard (PID)
Created: 11/16/2010 09:05:51 am
Last Modified: 11/16/2010 09:05:51 am

	Symbol	Var Type	Data Type	Comment
	EN	IN	BOOL	
LW0	PV_I	IN	INT	Process Variable Input: Range 0 to 32000
LD2	Setpoint_R	IN	REAL	Setpoint Input: Range 0.0 to 100.0
		IN		
		IN_OUT		
LW6	Output	OUT	INT	PID Output: Range 0 to 32000
		OUT		
LD8	Tmp_DI	TEMP	DWORD	
LD12	Tmp_R	TEMP	REAL	
		TEMP		

This POU was created by the PID formula of the S7-200 Instruction Wizard.
To enable this configuration within the program, use SM0.0 to call this Subroutine from the MAIN program block every scan cycle. This code configures PID 0. See DB1 for the PID loop variable table starting at VB100. This subroutine initializes the variables used by the PID control logic and starts the PID Interrupt "PID_EXE" routine. The PID interrupt routine is called cyclically based on the PID sample time. For a complete description of the PID instruction see the S7-200 System Manual. Note:When the PID is in manual mode the output should be controlled by writing a normalized value(0.00 to 1.00) to the Manual Output parameter instead of changing the output directly. This will automatically provide a bumpless transfer when the PID is returned to automatic mode.

Labvj3_PIDEXE / PID_EXE (INT1)

Block: PID_EXE
Author: S7-200 Instruction Wizard (PID)
Created: 11/16/2010 09:05:51 am
Last Modified: 11/16/2010 09:05:51 am

Symbol	Var Type	Data Type	Comment
	TEMP		
	TEMP		
	TEMP		
	TEMP		

This POU was created by the PID formula of the S7-200 Instruction Wizard.
This interrupt routine implements Timed Interrupt for PID execution. This interrupt routine was attached in subroutine "PID0_INIT"

Labvj3_PIDEXE / USR1

Symbol	Address	Comment
PTRU	VD64	Pointer No. 1 (controller output)
PTRY	VD68	Pointer No. 2 (process output)
cnt	VW72	Counter for saving data
PV_INPUT_VAL	VW16	Process variable from Analog IN
M_OUT_VAL	VW28	Controller output for Analog Out
MX	VD128	

Labvj3_PIDEXE / POU Symbols

Symbol	Address	Comment
SBR_0	SBR0	POU Comment
SBR_1	SBR1	POU Comment
PID0_INIT	SBR2	This POU was created by the PID formula of the S7-200 Instruction Wizard.
INT_0	INT0	POU Comment
PID_EXE	INT1	This POU was created by the PID formula of the S7-200 Instruction Wizard.
MAIN	OB1	Testing integrated PID function

Labvj3_PIDEXE / PID0_SYM

Symbol	Address	Comment
PID0_D_Counter	VW136	
PID0_D_Time	VD124	Derivative Time
PID0_I_Time	VD120	Integral Time
PID0_SampleTime	VD116	Sample Time (To modify, rerun the PID Wizard)
PID0_Gain	VD112	Loop Gain
PID0_Output	VD108	Calculated, Normalized Loop Output
PID0_SP	VD104	Normalized Process Setpoint
PID0_PV	VD100	Normalized Process Variable
PID0_Table	VB100	Loop Table Starting address for PID 0